3.5.2 Device Process Variation

To assess the sensitivity to variations in device parameters, statistical Monte-Carlo simulations have been performed. The test circuit is a XOR tree built out of 2-input XOR gates. Such a tree has n inputs and $\log n$ logic levels and computes the n-bit parity function.

The scatter plot in Figure 42 shows the correlation between the nominally identical latencies of two identical 32-b XOR tree designed in a 0.35μm technology. The distance of any sample from the unit line is due to statistical variations in the manufacturing process. The main result here is that while the absolute process variation is large the relative mismatch is at most 28ps as shown in the closeup view on the right in Figure 42. Given the fact the the maximum achievable frequency in this process is ≈2GHz or 500ps cycle time the delay variation due to mismatch is only 5.6% of the cycle time.

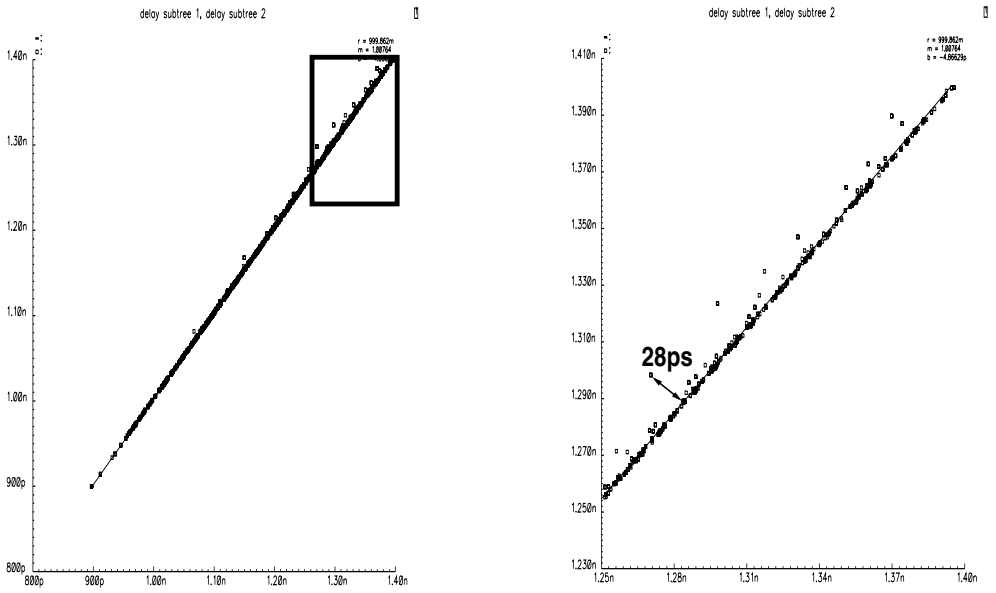


Figure 42. Monte-Carlo simulation of two 32-b XOR trees: process variation and device mismatch.

It is well known that in more advanced technologies the intra-die variations have a greater share on the total variation. In order to verify that AWP operation is still feasible in deep submicron CMOS technology Monte-Carlo simulations with a 130nm technology have been performed. This time 64-b XOR trees with six gate levels are used as test circuit. Figure 43 shows distributions of the delay variations after the first til sixth stage ($d1\_27 \ldots d6\_27$). The scatter plot in Figure 44 shows the variation after six gates at the output of the trees. From 1000 samples are 32 off grid; the remaining 96.8% of the samples show a delay variation of less than 10ps or 7.14% of the cycle time.

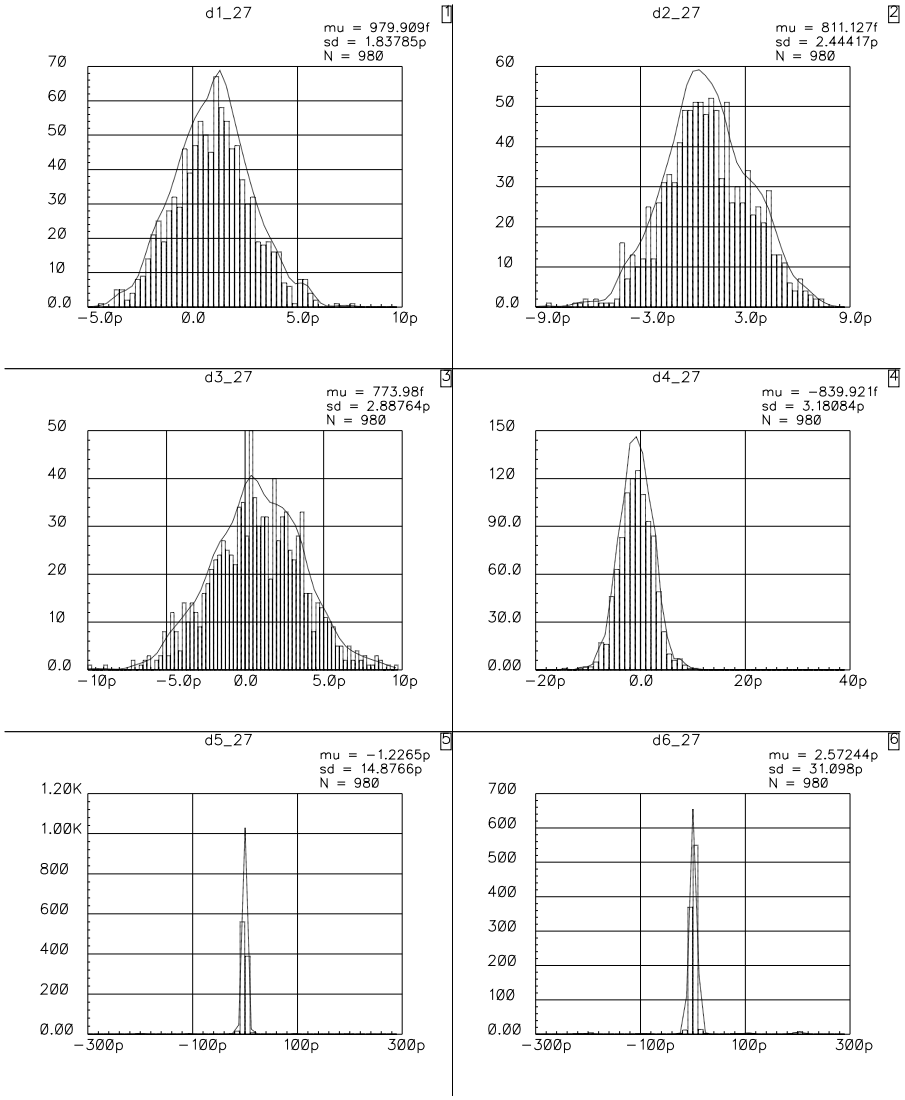


Figure 43. Histogram of Monte-Carlo simulation of two 64-b XOR trees: delay variation in the different levels.

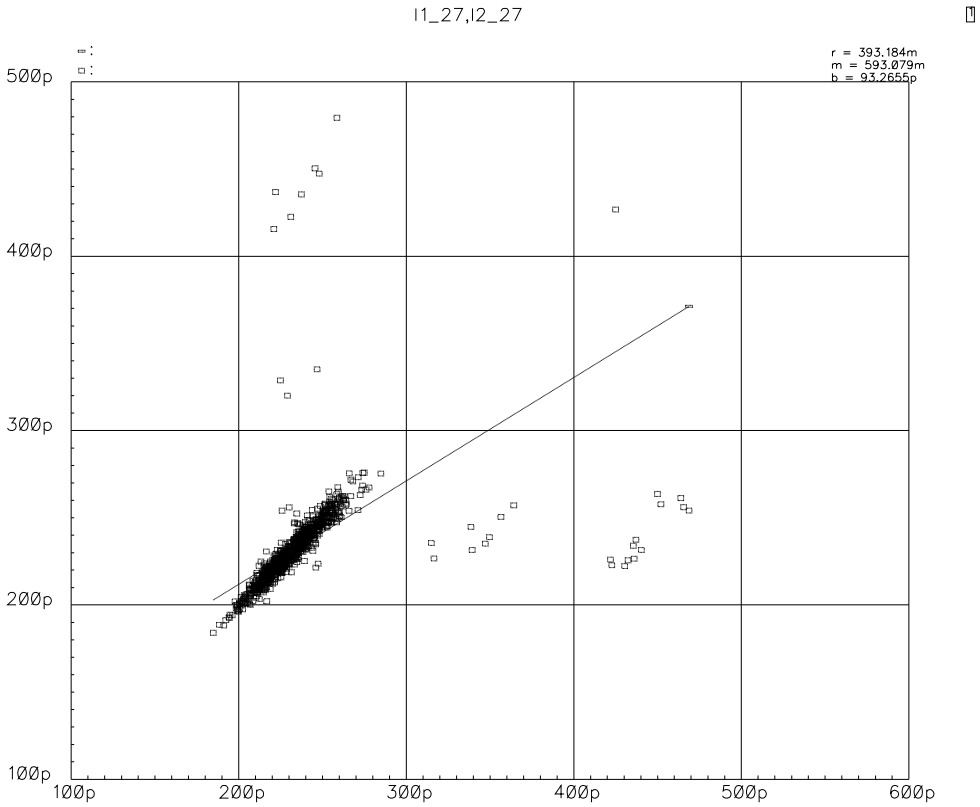


Figure 44. Scatterplot of Monte-Carlo simulation of two 64-b XOR trees: process variation and device mismatch.

For comparison the correlation of the latencies of two independent chains of 14 inverters each is determined via Monte Carlo simulation. Again histograms and scatter plot are given in Figure 45 and Figure 46. The scatter plot shows that the delay variations of the inverter chains are in the same range as the XOR trees. This is remarkable because in the latter are $O(2^n)$ devices involved with n being the depth, whereas the inverter chains

have only $O(n)$ devices. One would expect the XOR tree to have significantly larger delay variation due to its exponential device count. The reason is that the NMOS NAND stacks in the AWPCMOS gates perform some sort of "microscopic synchronization" of the data: as the pulldown occurs only after *both* inputs are over threshold delay variations from either input cannot propagate through unhindered; the probability at the output depends on the *product* of the input probabilities. The inverter chains, on the other hand, propagate delay variations very well.

Figure 47 and Figure 48 show corresponding simulations of XOR tree and inverter chain in current 90nm CMOS technology. It is obvious that AWPCMOS delays are well behaved also in an advanced technology.

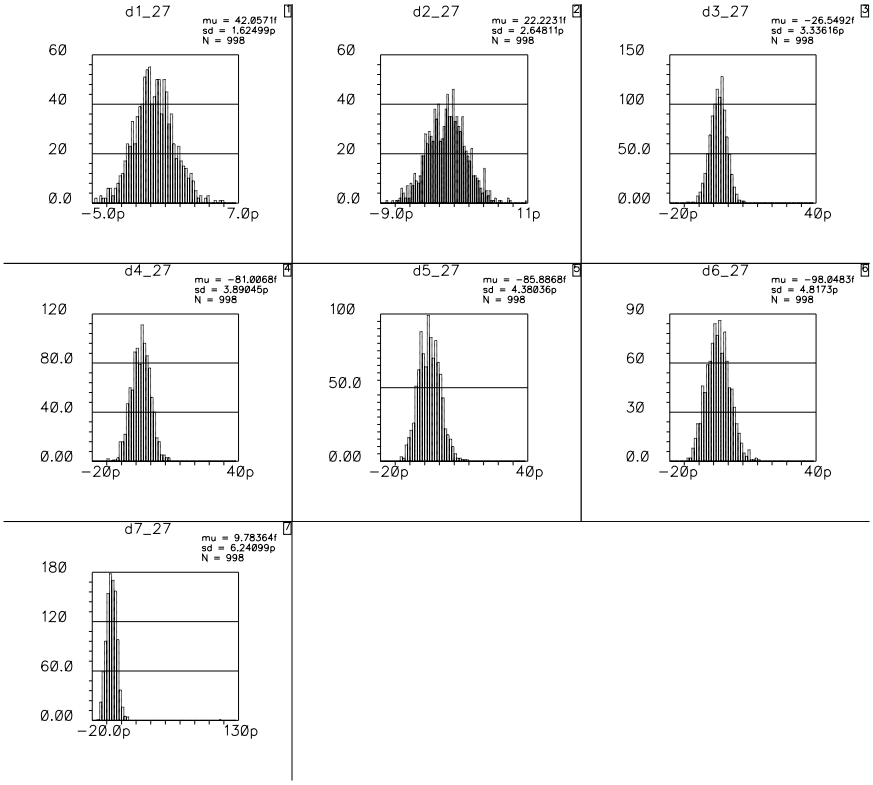


Figure 45. Histogram of Monte-Carlo simulation of two inverter chains: delay variation in the different levels.

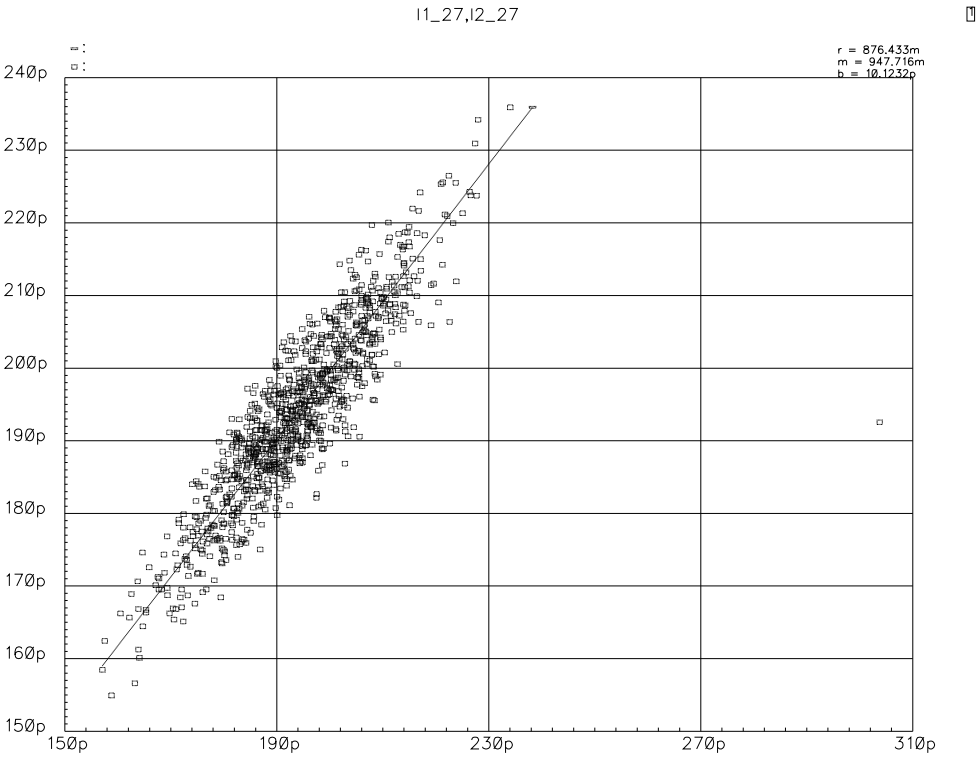


Figure 46. Scatterplot of Monte-Carlo simulation of two inverter chains: process variation and device mismatch.

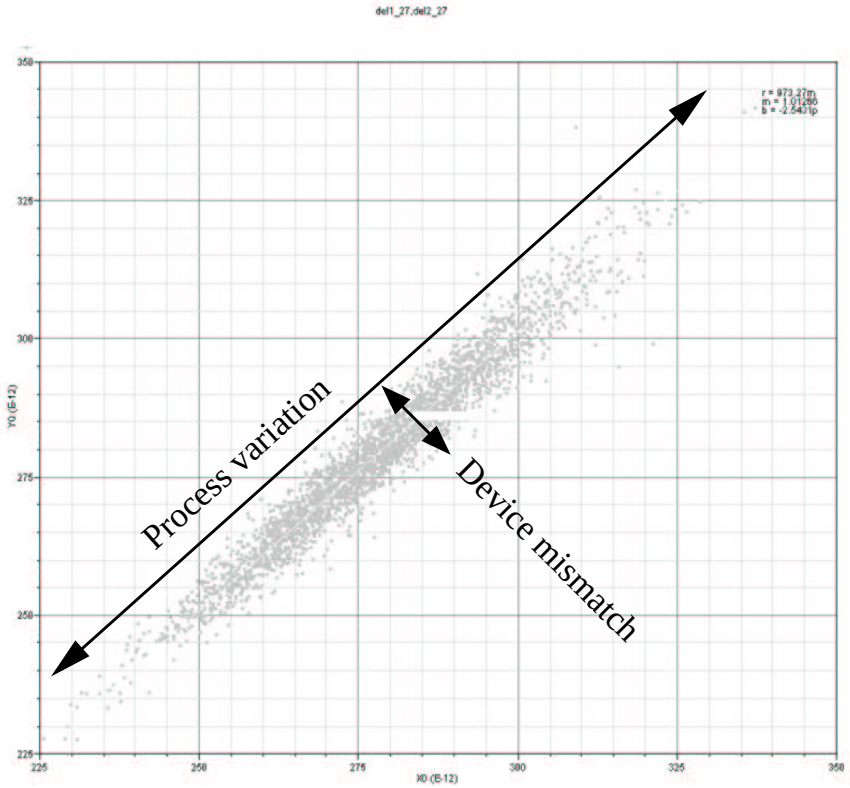


Figure 47. Monte-Carlo simulation of 64-b XOR trees in 90nm CMOS.

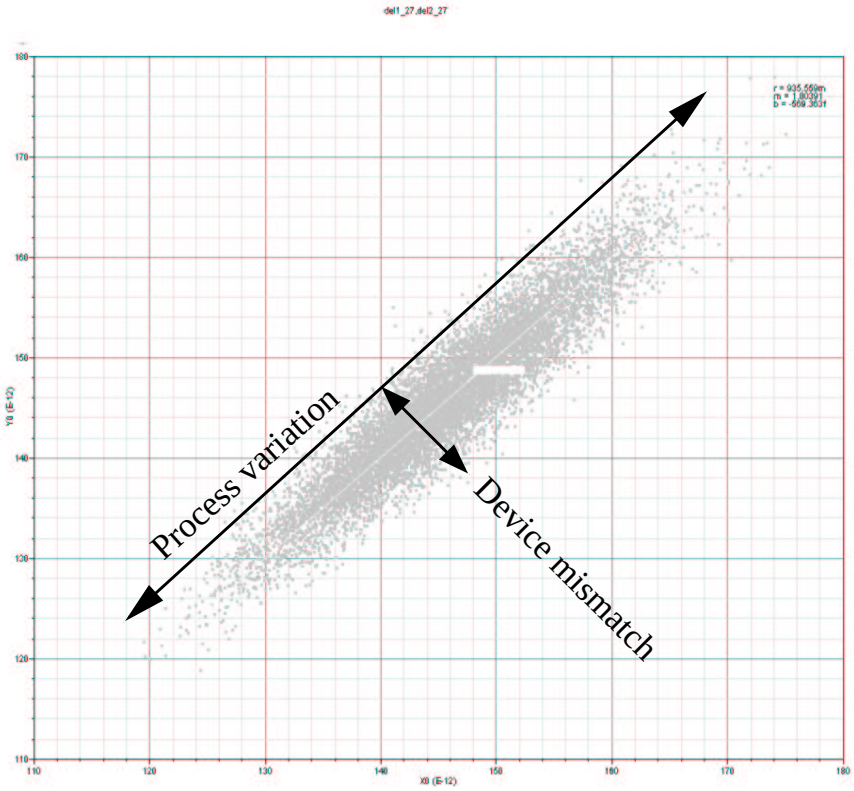


Figure 48. Monte-Carlo simulation of 14-inverter chain in 90nm CMOS.

3.6 Power Analysis

We have seen that AWPCMOS, due to its dynamic dual-rail nature, has high power consumption. When comparing with clocked synchronous pipelines, however, one must take into account the synchronous power overhead due to clocking and latching which is significant for fine-grain pipelines. A general comparison is difficult because the power figures depend on the specific circuit. Therefore an experiment was performed, again with our XOR tree as an example.

To assess the power efficiency of an AWPCMOS pipeline with respect to a conventional dual-rail domino pipeline at a given frequency, we compare the power spent by the AWPCMOS pipe *per single wave* with the power spent by a *non-pipelined* combinational dual-rail circuit with identical device sizes computing the same function in Figure 49. The circuit is a 64b XOR tree at 1.0GHz. The upper half shows AWPCMOS current, input and output. The integral under the current trace gives the charge for a single sample as 1.812e-11As equivalent to 59pJ at 3.3V. The combinational dual-rail domino circuit is clocked giving rise to 1pF parasitics. Additionally, a 500μm/200μm driver is needed to drive the clock net at 1.0GHz. Shown in the lower half the charge transport for a precharge/evaluate cycle is 4.030e-11As. AWPCMOS saves 55% energy and decreases the peak current by 19.4%. Domino latency is 1.87ns so that a 1.0GHz domino pipeline would have two pipeline stages separated by one pipeline register. This pipe register incurs additional overhead irrespective of the pipelining method. Therefore the AWPCMOS energy and peak current savings will be even bigger when sequential storage elements are used and the clock net load is increased. AWPCMOS will be as well more energy-efficient as advanced methodologies as skew-tolerant domino, self-resetting and asynchronous circuits as these still have to clock every gate when using dynamic logic.

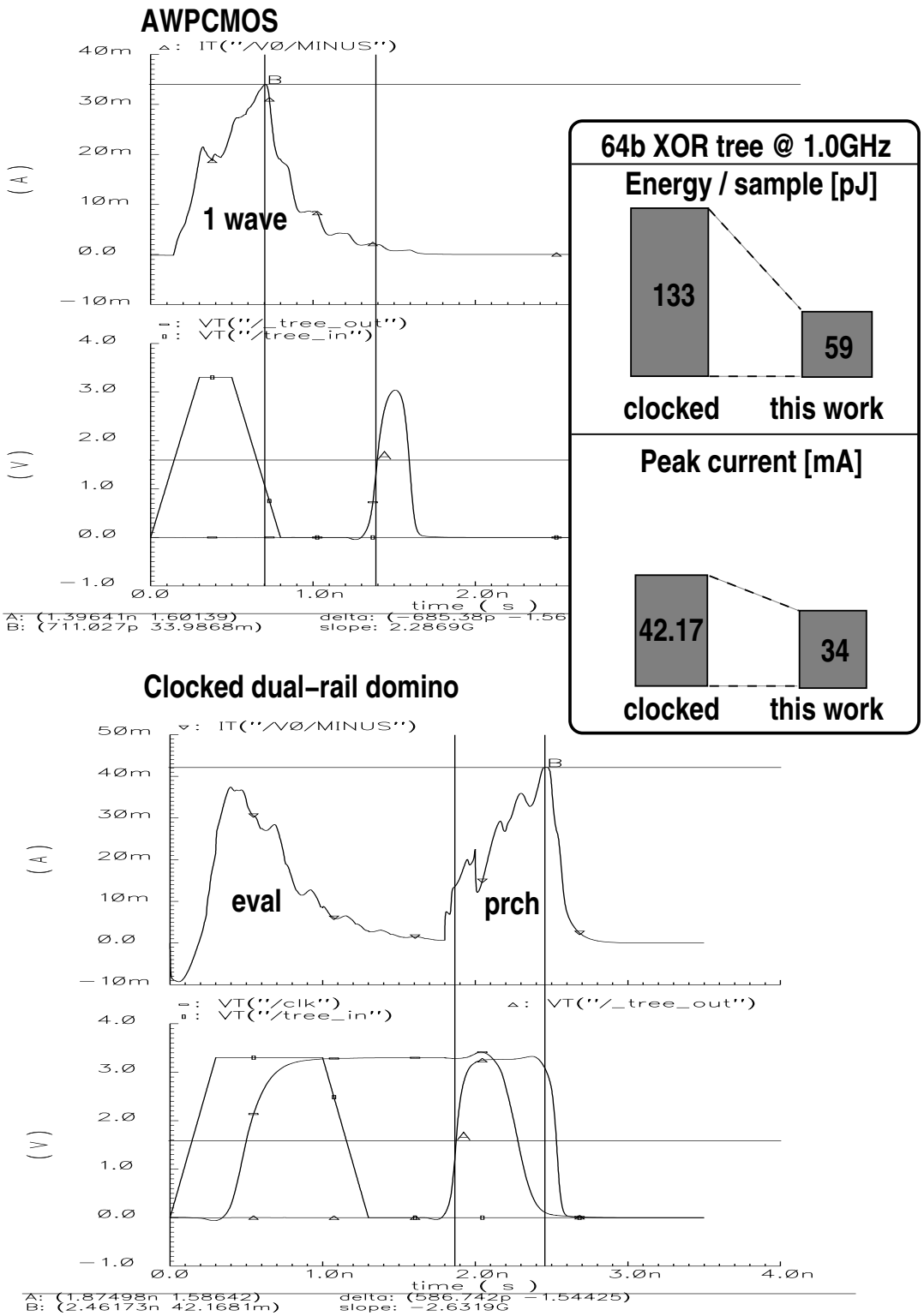


Figure 49. Power comparison AWPCMOS vs. clocked dual-rail domino.

3.7 Embedding the AWP into Synchronous Systems

The question arises as to which circuits the AWP method is best applied. Due to the importance of balanced logic paths the circuit should not be too big in order to mitigate intra-die variation. Furthermore, complete systems seldom need to run at the highest frequencies. Therefore the AWP is the best fit when small cycle time *and* latency are required at the macro level.

If some combinational logic has latency t_{lat} and is to be executed n times sequentially, the total latency t_{tot} for the job is just

$$t_{tot,seq} = n \cdot t_{lat}. \tag{3.18}$$

The same job executed on a pipeline with cycle time t_{cyc} takes

$$t_{tot,pipe} = t_{lat} + (n-1) \cdot t_{cyc}. \tag{3.19}$$

In any case the number of repetitions n is not negligible, otherwise the task wouldn't lend itself to pipelining in the first place. Minimizing t_{cyc} is thus paramount for minimizing $t_{tot,pipe}$. On the other hand, if n is not too big or can even be one in some cases, then minimizing t_{lat} matters as well. The simultaneous minimization of both cycle time and latency can be achieved best with the AWP.

The next question is how to feed the pipeline at very high rate and consume the results accordingly ? And how to embed the AWP in a robust manner into a larger system ?

In the simplest case an AWP would be plainly dropped in a synchronous system with the clock being used to launch data into the pipeline. If the output were to be latched with the same clock the scheme would suffer from the same drawbacks as synchronous wave pipelines. Furthermore, if not the whole system is likely to run at wave pipeline rate, how is the input clock generated from the system clock and where does the data come from ?

A better scheme to embed an AWP into a system, either synchronous or asynchronous, is to decouple it somewhat from its environment. This is necessary to sustain the required data rate for the pipeline, and it eases output synchronization. The general scheme looks like Figure 50.

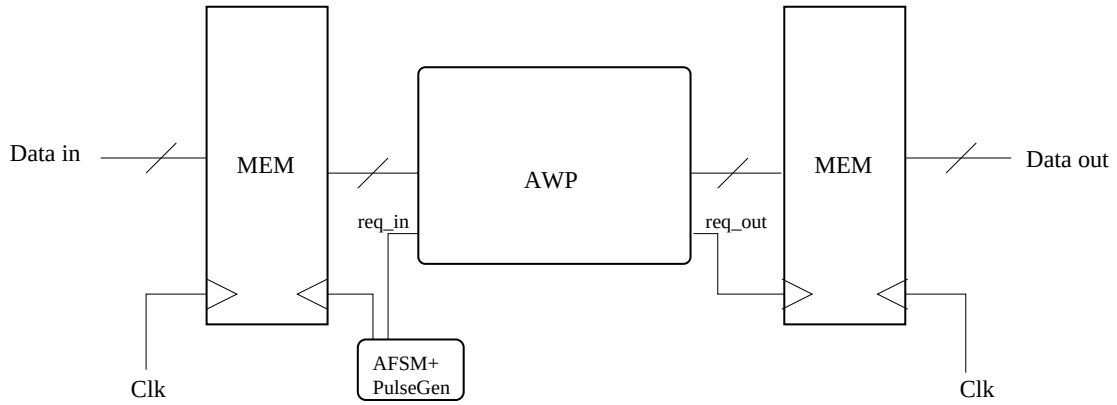


Figure 50. Embedding of AWP into synchronous host system.

Data is synchronously loaded into some memory structure. Thereafter, the synchronous clock is gated and an asynchronous state machine generates high-speed pulses to feed the data into the AWP preferrably using the scheme of Figure 28. Results are collected again

in some memory structure. When there is no more data to process for the pipeline the pulses will cease. The synchronous clock takes over then and loads the results back into the synchronous host system. Depending on the application the memory structures mentioned will be some shift register or memory with multiplexers capable of high data rates.

Let's consider the Advanced Encryption Standard (AES) Algorithm as an example. This is the symmetric block cipher replacing the venerable Data Encryption Standard (DES) algorithm. The AES operates on a 4 by 4 byte array called *State*. The most important and costly step in the algorithm is the *SubBytes* operation involving a nonlinear SBox. The same SBox is applied independently to all 16 bytes of the state. If only a single SBox were available SubBytes would take at least 16 clock cycles. This would already assume a pipelined SBox circuit as its latency normally doesn't fit in a single clock cycle. If the design is standard-cell it cannot take advantage of a faster running clock. Conventional architectures therefore use several SBox circuits to enhance throughput.

Using an AWP for the SubBytes operation as depicted in Figure 51 offers an unique opportunity to get along with a single SBox yielding throughput equivalent to a conventional architecture using a multitude of SBoxes. This is especially attractive for area constrained environments like smartcards.

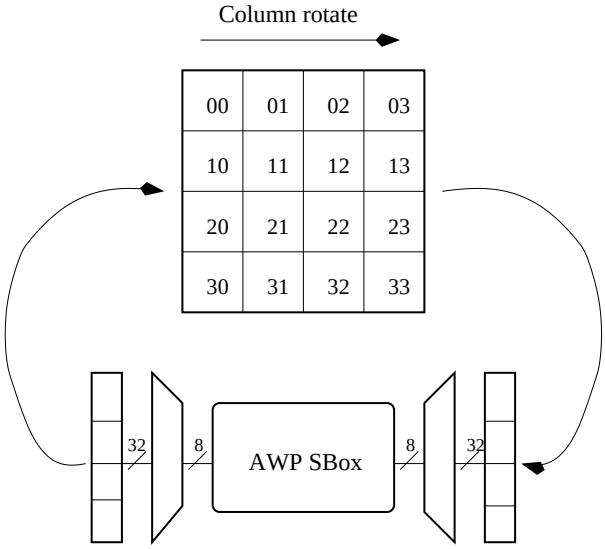


Figure 51. Using AWP for AES SubBytes operation.

We assume the state has been loaded by the preceding step. For the SubBytes step an AFSM (not shown here) generates a local high-speed clock to rotate the state column-wise and multiplex a 32-b word into 4 bytes. At the output of the pipeline a demultiplexer populates a word which is written back into the state.

With n being 16 in equation (3.19) for this example this is actually a case where latency matters as well. Let the latency of the SBox be 10ns and their cycle time be 2ns. Then without pipelining the whole SubBytes step would take $16 \cdot 10 = 160$ns. With pipelining we would have the total latency of SubBytes as $10 + 15 \cdot 2 = 40$ns, a speedup of four!

3.8 Summary

In light of the discussion of related work in chapter 2 and the description of AWPCMOS in this chapter, the distinguishing feature of AWPCMOS central to the contribution of this thesis can be stated as follows: it exhibits true wave pipelining using dynamic circuits and yet avoids the clock signal. As was pointed out in section 3.3 static circuits have inherent disadvantages due to level based logic when it comes to balancing delays. Dynamic circuits normally need a clock for every gate, not just sequential elements. Self-resetting circuits replace the global clock infrastructure by a local reset timing chain but still need a clock-like signal for every gate. The same is true for all asynchronous pipelines schemes as well. Only AWPCMOS has fully local precharge generation and does not need a clock signal. Only in AWPCMOS the cycle time is always six inverter delays independent of logic depth. In Figure 52 the clock nets for an AWPCMOS pipeline and for some other pipeline scheme using dynamic circuits are contrasted diagrammatically.

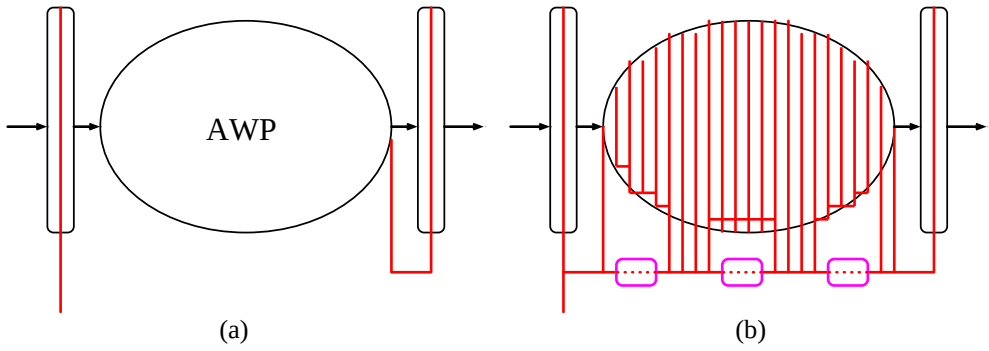


Figure 52. Highlighting clock nets in AWPCMOS (a) vs. other schemes (b).

Clock nets are drawn red. In an AWP only input and output registers are clocked, which represents the absolute minimum. In a dual-rail AWP the output clock signal can be derived in the logic itself as was shown in section 3.4.4 and is depicted in Figure 52 (a). The AWPCMOS pipeline core logic does not have a clock.

In contrast, in Figure 52 (b) the generic configuration of *any* other scheme is shown, be it dynamic logic with global clock or self-resetting or asynchronous pipeline. The implementation details of sequential elements don't matter as well. Rather, the important issue is that dynamic logic needs *every* gate supplied with a clock-like signal. There are basically three methods for clock distribution symbolized by the purple boxes: first, in a global clocking the boxes are empty; second, in schemes that propagate the clock along the datapath, the boxes will contain buffers that amplify and delay the clock; third, in asynchronous handshake schemes, the boxes contain the local handshake control as well.

It is obvious that AWPCMOS presents significantly less clock load. This has two advantages: first, power is saved. The high switching activity of a dual-rail AWP together with the cost for delay padding must be taken into account, however, to determine the net power savings. Second, any clock-like control signal, even in asynchronous handshake schemes, has to obey a setup/hold constraint. This necessitates timing margins and negatively impacts latency and cycle time. These culprits are absent in AWPCMOS.

Chapter 4

64-Bit Pipelined Parallel Adder

4.1 Overview

THIS chapter describes the implementation of a 64-bit carry look-ahead adder in AWPCMOS. It shows that Nanosecond cycle times can be achieved on wide data paths even in a 0.6μm CMOS technology. There have been several wave-pipelined adders realized so far, however the one described here is the first 64-bit wide. The testchip is called GATS for Gigahertz adder test site in resemblance to IBM's Gigahertz unit test site, guTS [S+98]. Table 2 gives data for some adders described in the literature[1].

Author	Adder	Technology	t_{pd}	f_{max}	Area	Power
Liu et al. [LGF+94]	16-bit KS	2μm 2M	38.8ns	250MHz	13.27mm^2	3W
Hwang et al. [HGS+99]	64-bit KS	0.25μm 5M	1.5ns nom	444MHz	0.44mm^2	299mW @ 300MHz
Ruiz [Ruiz98]	32-bit KS	1.0μm 2M	12ns	83.3MHz	2.31mm^2	79.3mW @ 10MHz
Tang et al. [TCB+00]	16-bit KS	AMS 0.6μm	6ns	370MHz	7.29mm^2	n.a.
Morinaka et al. [MMN+95]	64-bit CS	0.5μm 3M	2.6ns	385MHz	0.27mm^2	33mW @ 200MHz

Table 2. Characteristics of previously realized adders.

This chapter is organized as follows: first, the Kogge-Stone carry scheme used in GATS is described; following, the AWPCMOS gates used in the adder are detailed. Then, the on-chip high-speed test circuitry is explained. Finally, the problems encountered on silicon and measurement results are presented.

4.2 Brent-Kung/Kogge-Stone Adder Architecture

For a realization in AWPCMOS an adder architecture is desirable which features balanced logic path delays without the need for much padding and which has limited fanin and fanout. For non-pipelined adders the required hardware is related to the carry evaluation scheme. A simple ripple adder consists just of n full adders, i.e. $O(n)$ hardware, whereas a sophisticated carry look-ahead adder uses $O(n \log n)$ hardware. This is in contrast to fine-grain pipelined adders where the hardware cost is always $O(width \times latency)$ and may be dominated by padding, i.e. deskewing buffers.

A simple ripple carry adder with latency n can in principle achieve short cycle times but uses only n productive gates, i.e. full-adders, compared to a total of n^2 gates used for n-bit

[1] KS means Kogge-Stone architecture; CS denotes carry select.

inputs. The case $n = 64$ results in $1 - \frac{1}{64} = 98.4\%$ of the gates wasted for padding which is clearly inappropriate. This situation is depicted in Figure 53 which shows the diagonal of the adder only being populated by full adders.

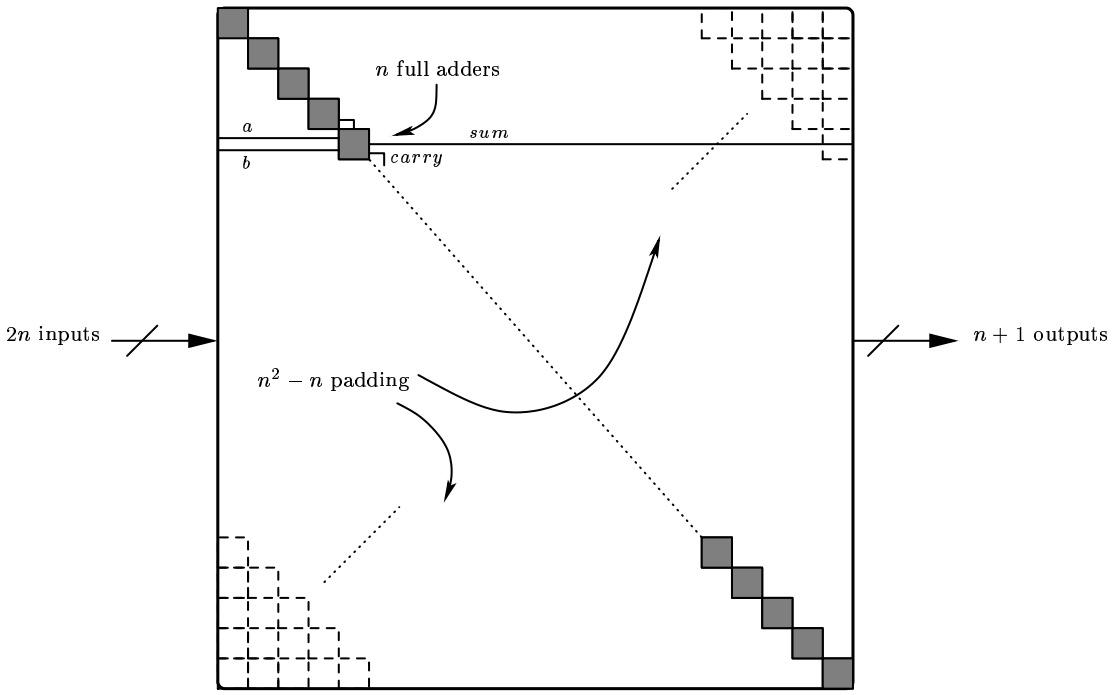


Figure 53. Pipelined ripple-carry adder.

As the width n of the adder is fixed the only way to reduce the amount of hardware is to use a scheme with reduced latency. Therefore a carry look-ahead tree structure based on recurrence equations is the best candidate as the latency of $O(\log n)$ is optimal and the structure is regular. Furthermore, the number of deskewing buffers needed for delay padding is moderate; the amount of hardware is of order $O(n \log n)$ as in the non-pipelined case.

In [BK82] Brent and Kung describe an adder architecture that is the basis of modern carry look-ahead adders. The critical path is $2 + 2\log_2 n$ gates. It can be optimized to $2 + \log_2 n$ gates according to Kogge and Stone [KS73].

In such a scheme the first level consists of half adders computing the initial propagate and generate terms,

$$p_i = a_i \oplus b_i \ , \quad g_i = a_i \cdot b_i. \tag{4.1}$$

The carries are recursively computed in a tree structure. Let $P_{i,k}$ and $G_{i,k}$ denote the carry propagate and generate terms for subtrees ranging from bit position i to k, respectively. These are computed using the P and G terms of adjacent smaller subtrees with ranges $i \ldots j$ and $j+1 \ldots k$ as

$$P_{i,k} = P_{i,j} \cdot P_{j+1,k} \ , \quad G_{i,k} = G_{j+1,k} + G_{i,j} \cdot P_{j+1,k}. \tag{4.2}$$

The interpretation of the propagate P term in (4.2) is that the larger tree propagates the carry from bit i to bit k if the subtrees do so from position i to j *and* from position $j+1$ to k. The generate G term is true if either the upper subtree has a generate $G_{j+1,k}$ or the lower one generates a carry $G_{i,j}$ that is propagated by the higher subtree, $P_{j+1,k}$.

The equations given in (4.2) are for the radix 2 case and yield a carry tree with $\log_2 n$ levels and fanin/fanout of two. It can be modified for radix four resulting in $\log_4 n$ levels and fanin/fanout of four. As higher fanin/fanout increase the cycle time in AWPCMOS radix 2 was chosen for the adder.

Once all carries $c_i = G_{0,i}$, $0 \leq i \leq n$ have been computed the sum bits s_i can be readily computed using a final XOR as

$$s_0 = p_0 \tag{4.3}$$

$$s_i = p_i \oplus c_{i-1}\,, \quad i > 0. \tag{4.4}$$

From equations (4.3) and (4.4) it can be seen that the initial propagate terms are needed for sum generation and thus have to be buffered through the adder. Altogether the adder has a latency of eight gate delays.

4.3 Adder Building Blocks

4.3.1 Initial Propagate/Generate Gate

The circuit shown in Figure 54 is a straightforward implementation of the equations in (4.1). The XOR gate for p necessitates dual-rail inputs.

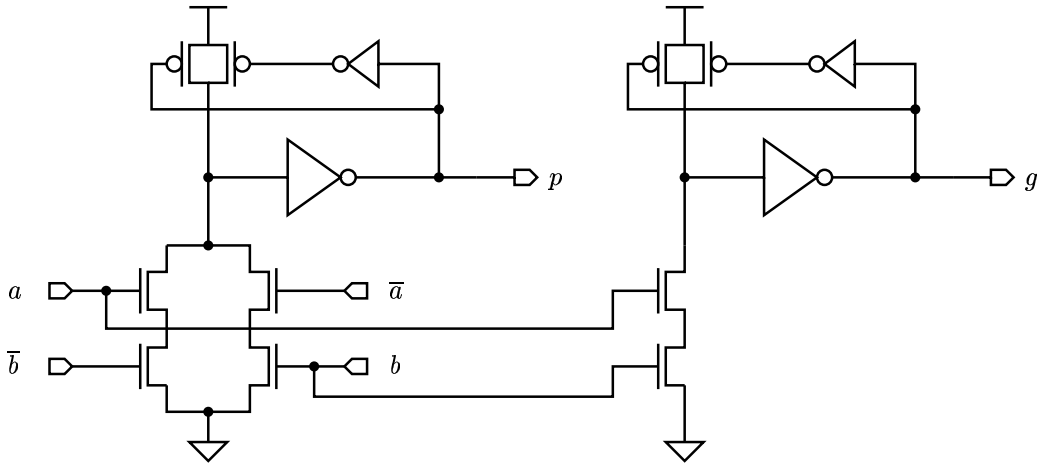


Figure 54. Initial propagate/generate gate.

4.3.2 Group Propagate/Generate/Buffer Gate

The circuit for equation (4.2) is shown in Figure 55. The gate for G_{ik} uses a padding device and the propagate buffer a weaker single NMOS to equalize pulldown delays. The fact that not both G_{jk} and P_{jk} can be one ensures that at most one pulldown path is active in the gate for G_{ik}. This is due to the recursive nature of equation (4.2) and the fact that $p_i = g_i = 1$ in (4.1) is impossible.

The whole carry tree consists of six levels of such gates, each level comprising 64 gates. It is a noteworthy example of a circuit that can be done using single-rail only.

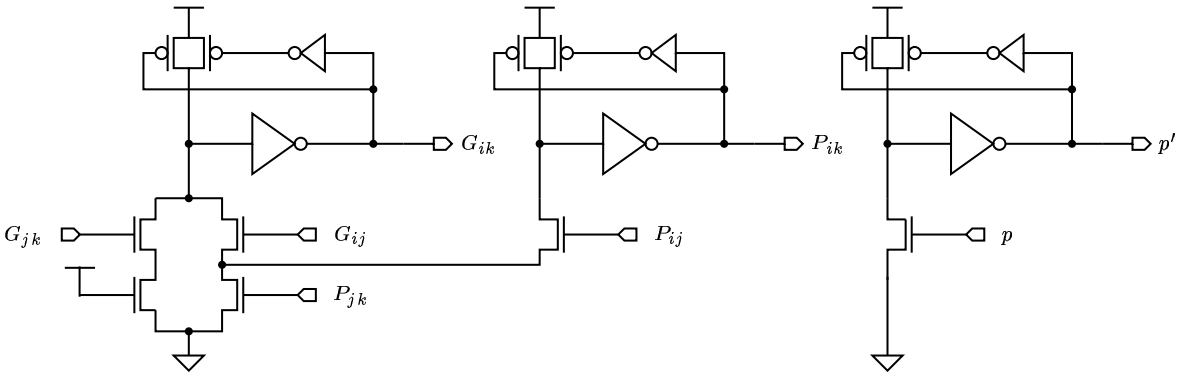


Figure 55. Group propagate/generate/buffer gate.

4.3.3 Final XOR

Figure 56 shows the XOR gate producing the sum bits. It is obvious that this gate with simple input inverters suffers from the monotonicity problem mentioned in section 3.4.2. It was not recognized when doing the first-time design and leads to unstable behaviour.

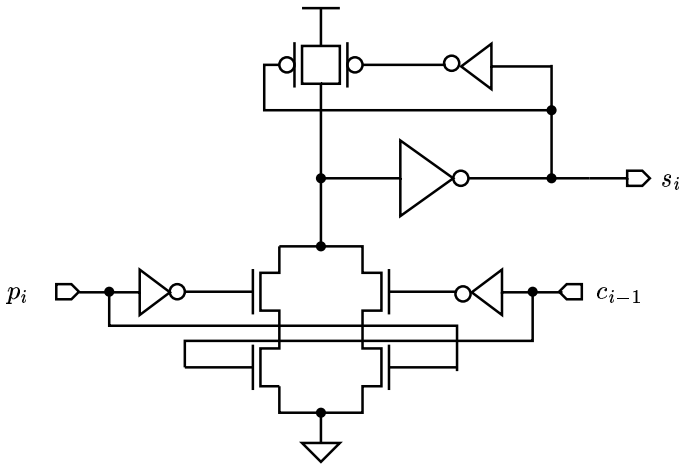


Figure 56. Sum generating XOR gate.

The testchip architecture consisting of the adder core and associated test environment are shown schematically in Figure 57. The characteristic carry tree interconnect scheme is apparent. A die photo of the GATS chip is shown in Figure 58.

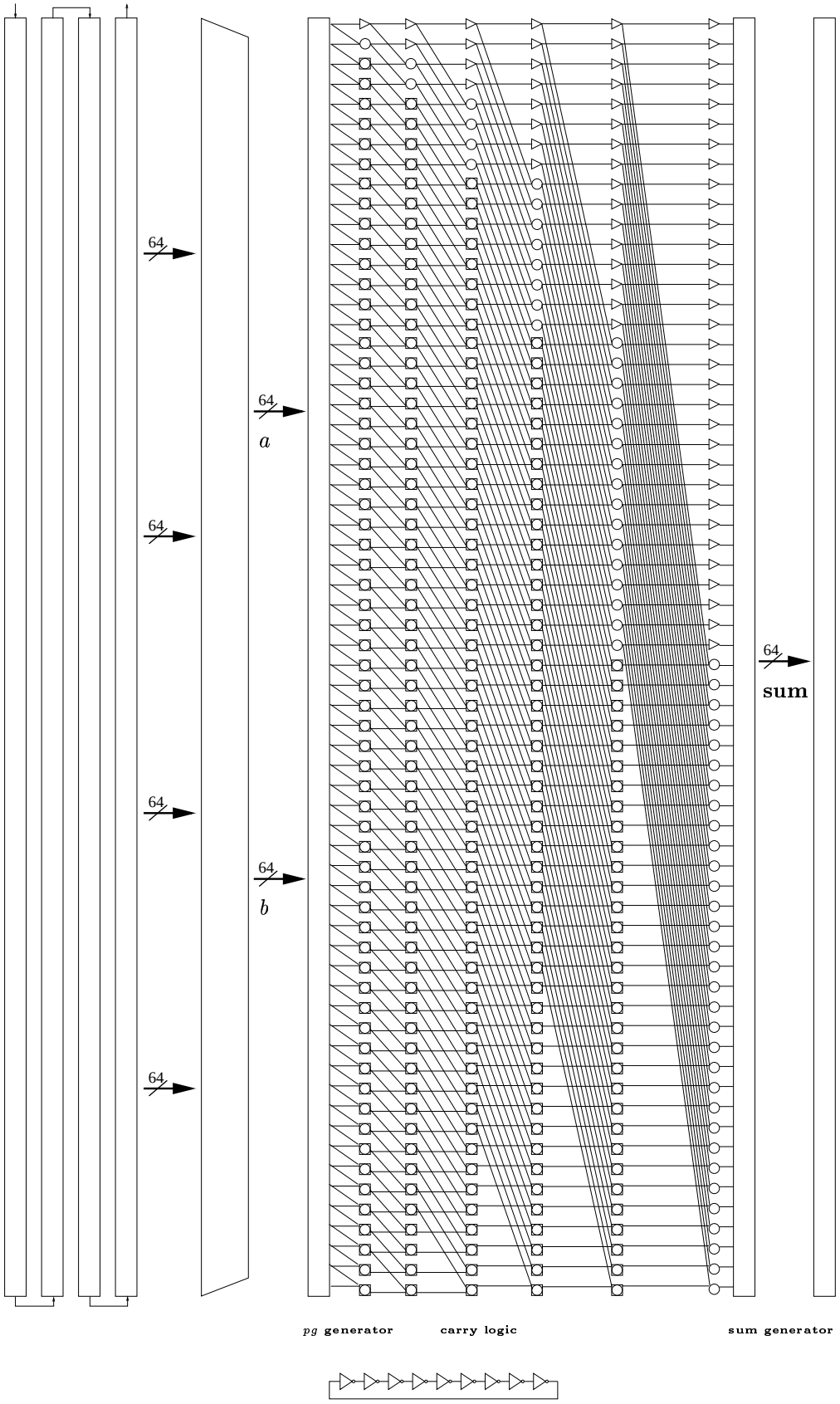


Figure 57. GATS architecture.

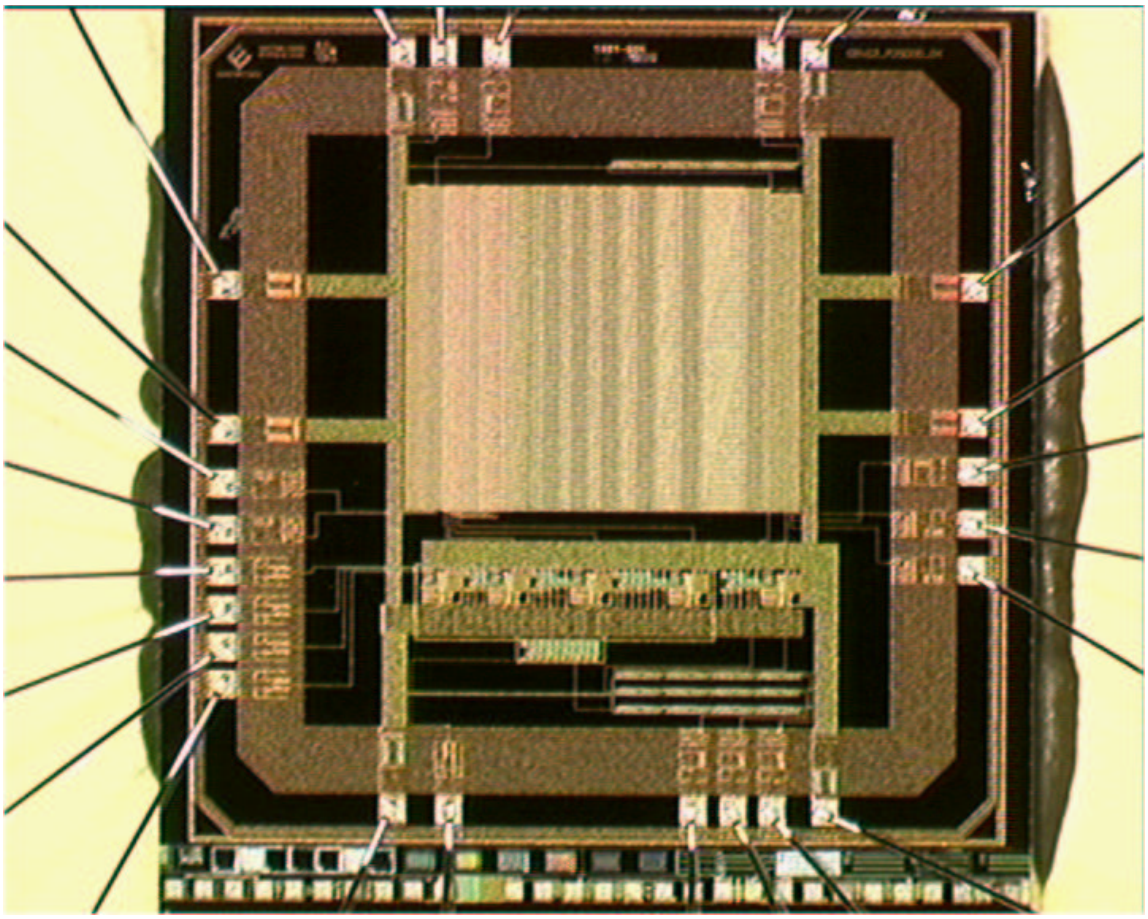


Figure 58. GATS die photo.

4.4 On-Chip Test Circuitry

The fundamental problem with high-speed circuits like the AWPCMOS adder described here is that the complexity of measurement and test rises dramatically. The reasons for this are twofold: first, at-speed test stretches the capabilities of the testing hardware; second, due to the limitations of ATE and chip pads it is mandatory to test the parts running at Gigahertz frequencies on chip. This calls for on-chip pattern generation that on the one hand can interact with the relatively slow tester hardware and on the other hand is able to stress the high speed circuits on the chip.

The basic idea to achieve this is described in [SCC+92] dealing with testing a wave-pipelined SRAM. Patterns coming from the tester enter the chip at a speed that the pads can handle through some sort of shift register. High speed pattern generation proceeds by combining in a parallel fashion the data that was serially loaded into the register before. This can be done by a multiplexor scheme. The key to high speed is that the pads are not involved and only a limited amount of the loaded data has to be moved to produce pattern vectors.

The on-chip test circuitry used in GATS is described in more detail below, rf. Figure 57. It will become apparent that the task of generating test vectors for a 64-bit adder at 1.0GHz in 0.6μm CMOS is challenging.

4.4.1 VCO and Pulse Generators

The source for pulses is a VCO built as a ring of nine current starved inverters. The frequency can be set by an analog bias via an external analog pad. The VCO has taps after the first, third, fifth, seventh and ninth inverter connecting to adjustable delay elements. The first four taps serve to overlay suitably delayed versions of the VCO output to yield 1.0GHz out of the nominal 250MHz VCO frequency. To this end pulse amplifiers generate strong pulses off the rising edges of the VCO taps. The last tap serves to clock the result register. Unfortunately, the timing of the four pulse phases is difficult to calibrate via the analog biases and works only for two out of four possible pulses.

4.5 Pattern Shift Register

Data enters the chip as a slow bit stream through the pad named serial_in. Non-overlapping clock phases φ_1, φ_2 are supplied by the tester. Instead of local clock generation both phases are connected to pads ensuring robust operation regardless of the slew rate of the tester inputs and input pad outputs. Schematic and layout of the basic shift register cell are shown in Figure 59 and Figure 60, respectively. The whole shift register is organized as four meanders each containing 64 bits.

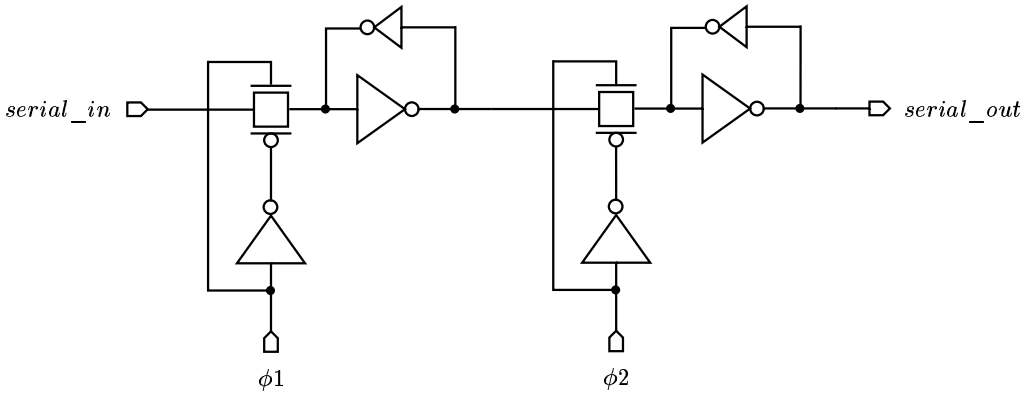


Figure 59. Shift register cell.

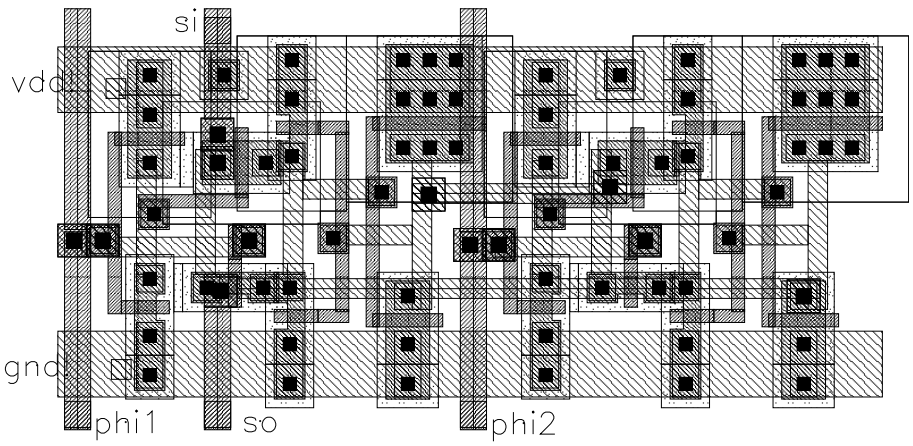


Figure 60. Shift cell layout.

4.5.1 Multiplexor-Based Pattern Generation

The multiplexor consists of 64 cells with schematic and layout shown in Figure 61 and Figure 62, respectively.

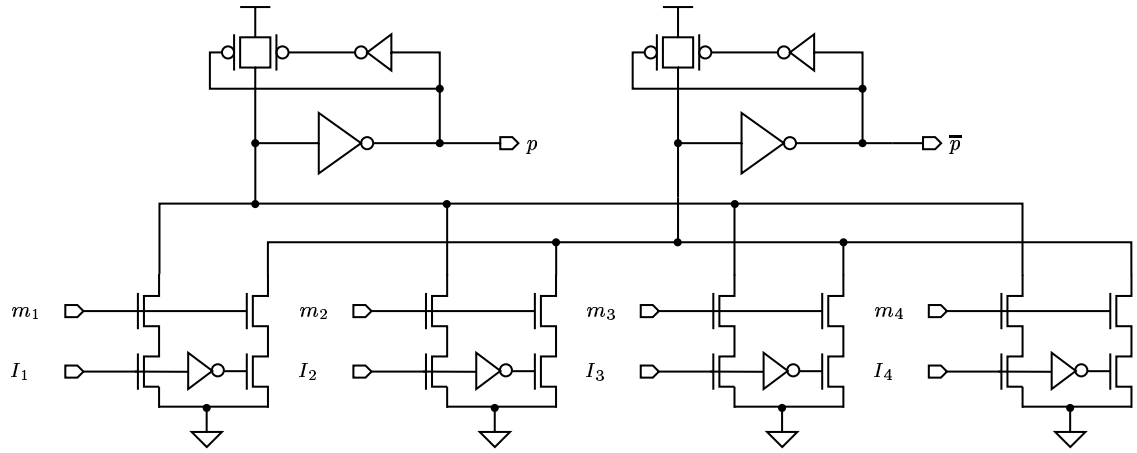


Figure 61. 4:1 multiplexor and dual-rail pulse generator.

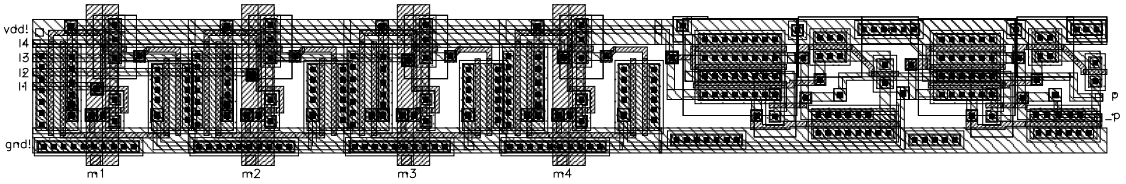


Figure 62. Multiplexor cell layout.

Each cell connects to the four shift cells lying in a row as shown in Figure 63. The multiplexors are one-hot controlled by the pulses m1 til m4. One full cycle therefore generates patterns $(a, b) = (s_2, s_1), (s_1, s_0), (s_3, s_2), (s_0, s_3)$.

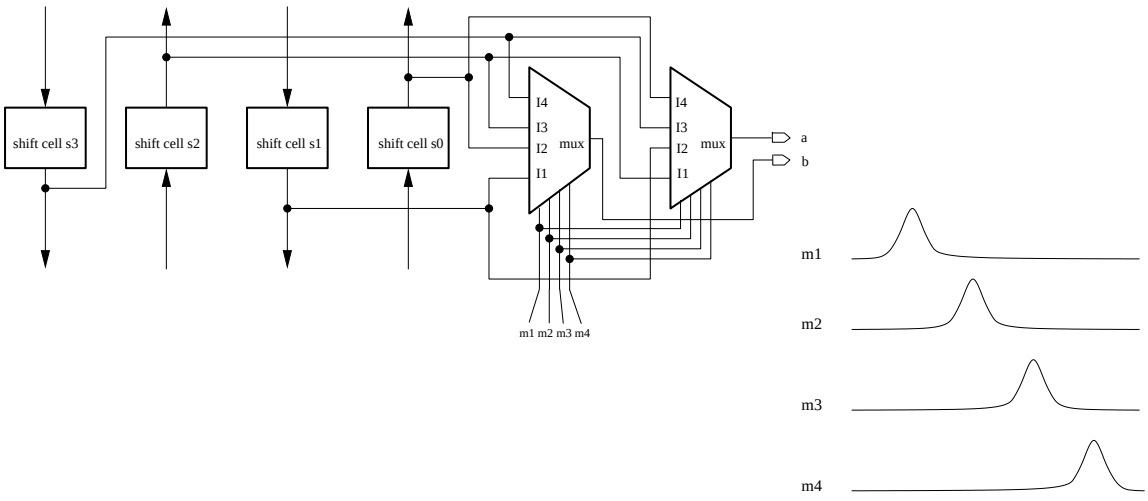


Figure 63. Slice wiring scheme for one-hot multiplexors.

4.6 Measurement Results

Apart from the mentioned design bugs concerning the conversion to dual-rail before the final XOR and the timing of the four delayed pulses controlling the pattern generation the device initially behaved very strangely on a HP82000 tester. As a simple sanity check the input shift register was set to feedthrough mode by $\varphi_1 = \varphi_2 = 1$ so that the serial_out pad should follow the input at serial_in pad. However, this was not the case. The problem was finally traced to the lack of clear reset of the pulse generation and the fact that the tester sets all inputs to one before the actual test sequence commences. This can leave the chip in a state where massive current is drawn which deteriorates Vdd.

The first oscilloscope trace in Figure 64 illustrates this situation where during tester initialization all inputs are set to one and then to zero. Setting to zero the analog bias voltages controlling the VCO and the delay elements can lead to cross current. This in turn violates the current limit imposed by the tester so that the regulator lowers the voltage. In about half of the cases Vdd is stuck at about 3V and serial_out does not react.

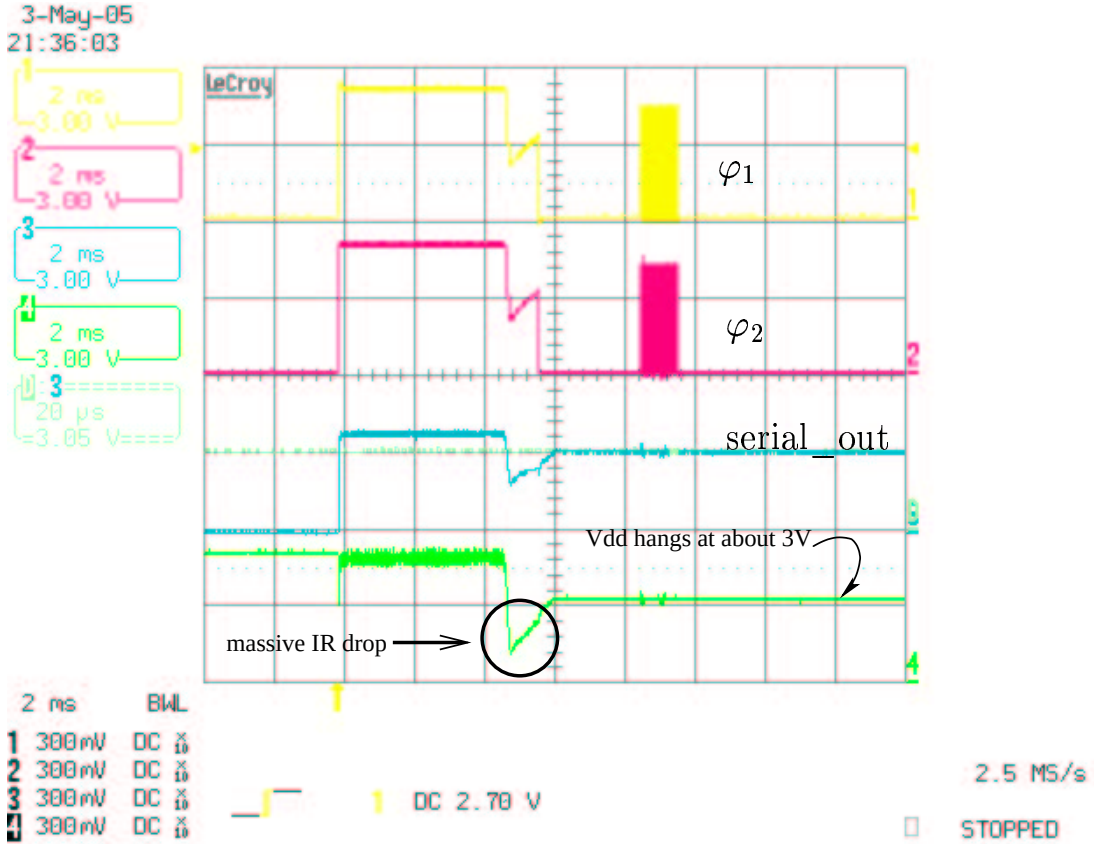


Figure 64. Voltage doesn't recover.

In the lucky cases, shown in Figure 65, Vdd happens to recover and the bits prviously shifted in show up correctly at serial_out as shown in the zoom inset.

A closeup of one lucky case is shown in Figure 66 where the non-overlapping clock phases supplied by the tester can be seen. According to Figure 59 the output serial_out changes with the rising edge of φ_2.

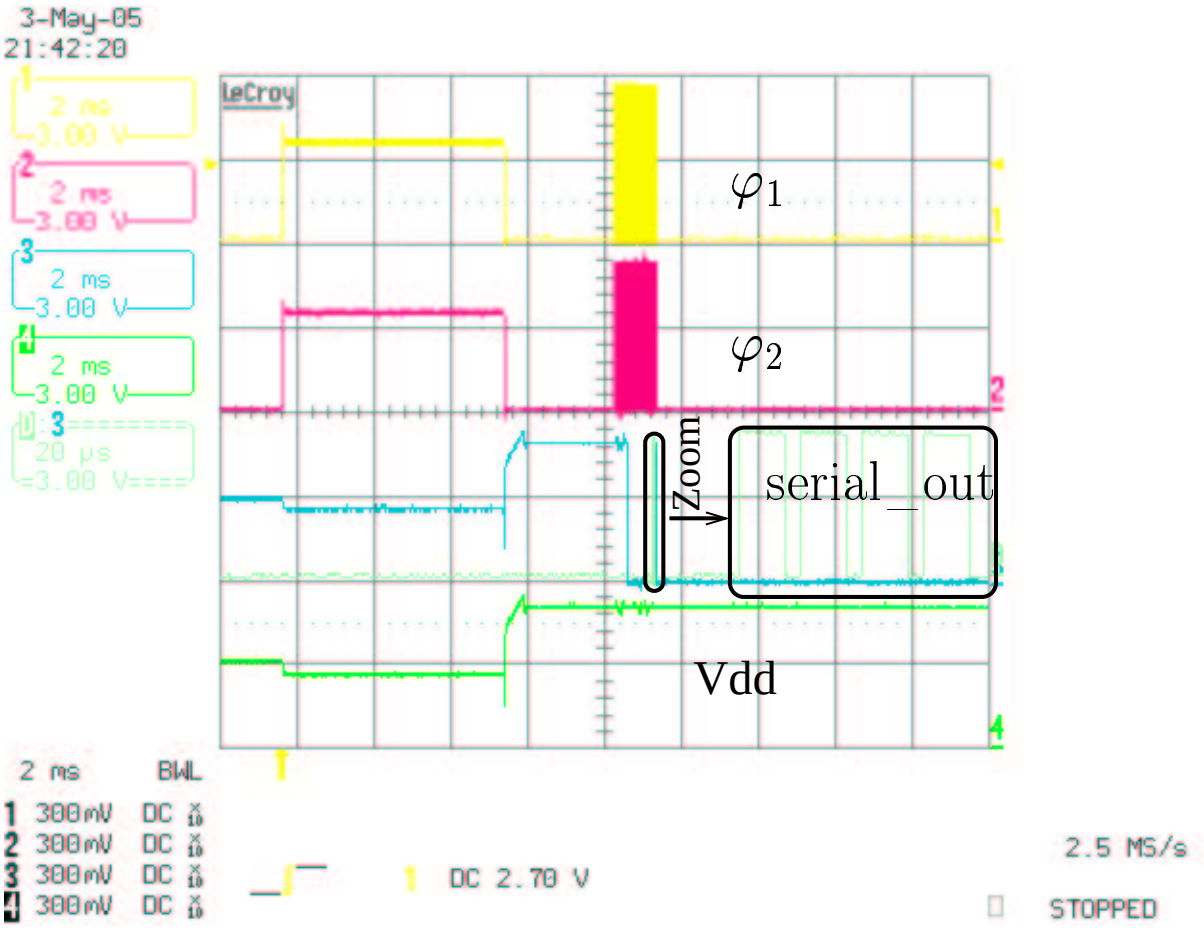


Figure 65. Voltage recovers.

The second and third trace in Figure 67 show output pad signals revealing a lot of noise generated in the power lines while the VCO is running. This is due to the lack of on-chip decoupling capacitance and a single power supply for the adder core, pulse generation, and I/O. Due to the above mentioned inadequacies in the design a functional test on the HP82000 was not possible. However, confidence in the device could be gained by some indirect means. The VCO output after the sixth inverter and the carry out of the adder have been wired to sampler circuits. These samplers employ 4 divide-by-2 elements in series thus effectively dividing the frequency by 16. This brings the frequency down to a value amenable for the pads.

Figure 68 shows the behaviour of the carry out signal. The shift register is again in feedthrough mode with serial_in being one. This is a covenient way to load only ones in a flash. The adder then does always

$$\begin{aligned} & 0xFFFFFFFFFFFFFFFF \\ + \quad & 0xFFFFFFFFFFFFFFFF \\ = \quad & 0x1FFFFFFFFFFFFFFFE, \end{aligned}$$

i.e. carry_out is one. With only m1 activated the carry should follow the VCO output which is affirmed by Figure 68. The VCO is running while the bias is active.

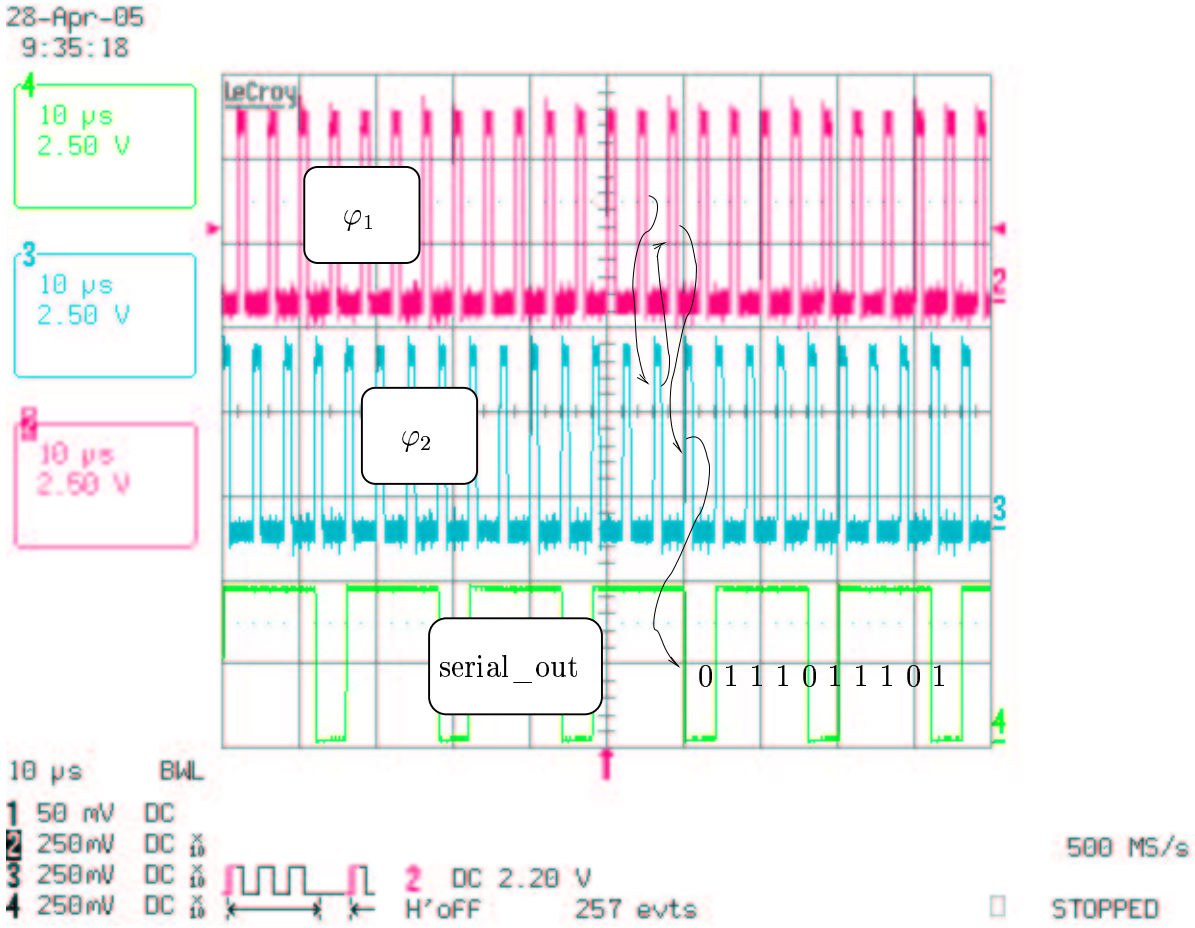


Figure 66. Shift register test in GATS.

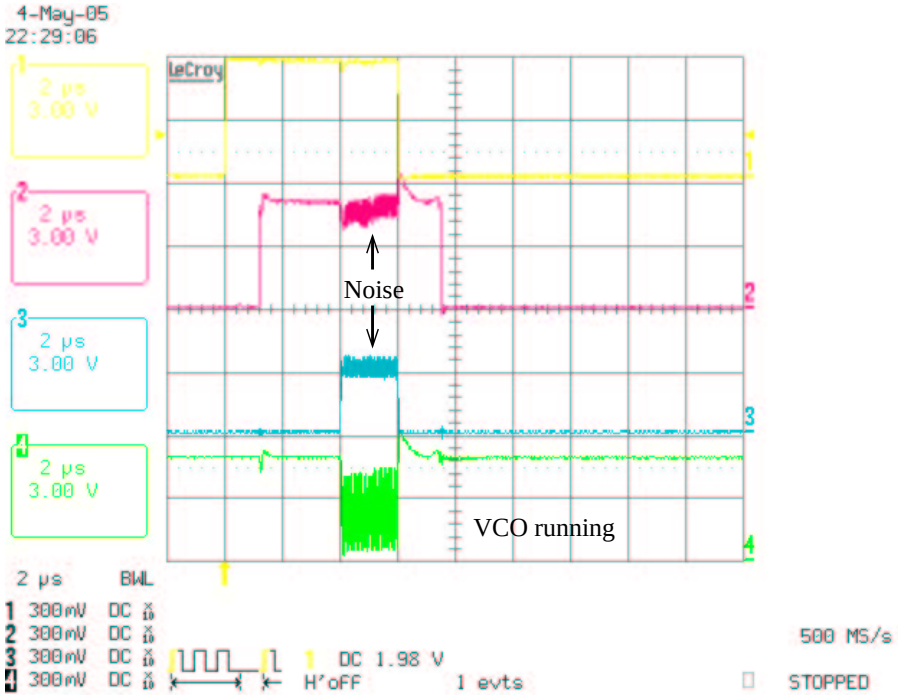


Figure 67. Noise generated while VCO is running.

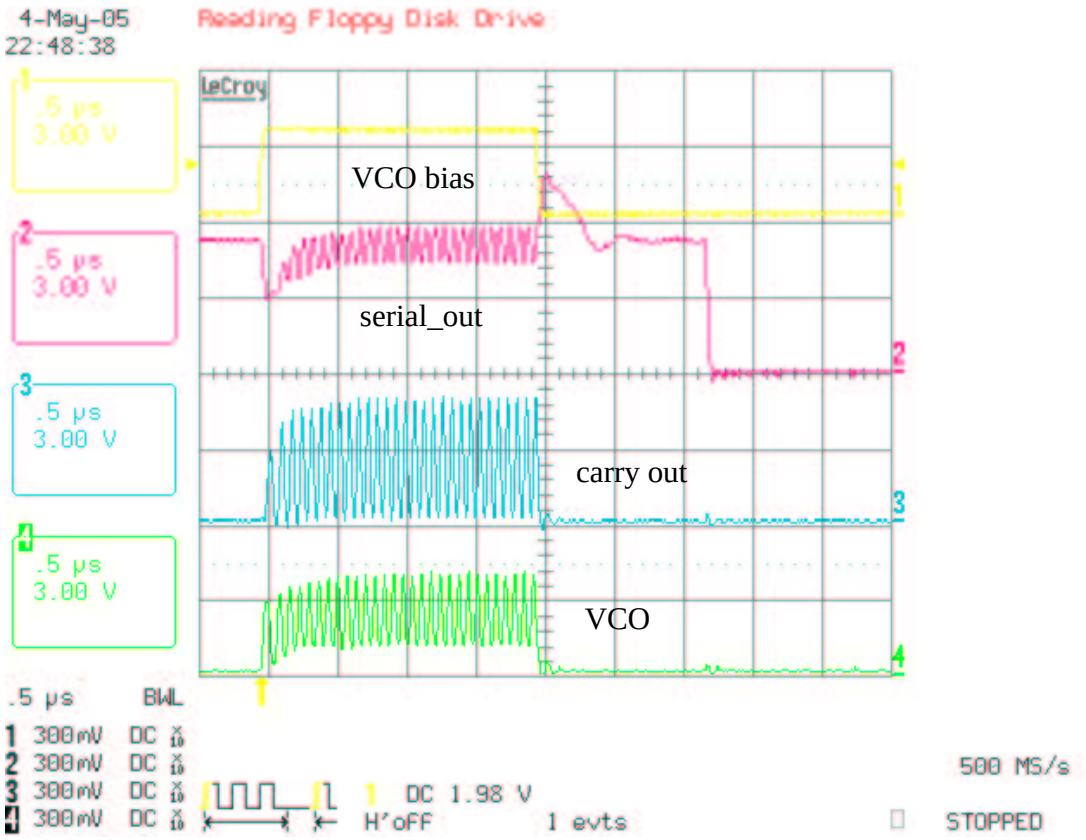


Figure 68. Carry out with one pulse active.

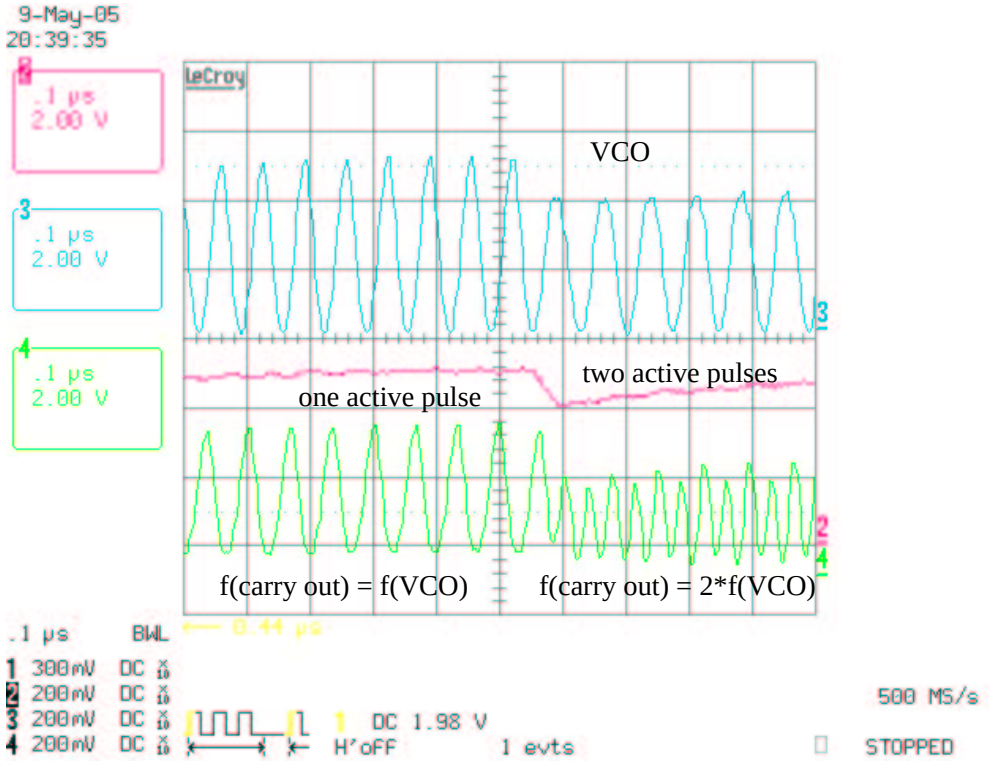


Figure 69. Carry out with two pulses active.

The correct function of the carry out requires correct working of the whole carry tree which is the main part of the adder. So the measurement is strong evidence that the adder core is healthy, i.e. matches simulations.

The final measurement in Figure 69 is similar but this time m1 and m2 are activated, i.e. carry_out shows double the VCO frequency. The middle trace is another output showing a dip when the second pulse is activated. The swings of the VCO and carry outputs are degraded probably due to speed limitations of the output pads. The internal frequency of carry_out at the sampler input is about 500MHz resulting at 31.25MHz externally.

4.7 Summary

This experiment has demonstrated that a Nanosecond cycle in a 64-bit datapath is feasible in a 0.6μm CMOS technology. However, one bug in the adder core and one in the pulse generation obviate full functional testing. Nevertheless, these problems are not a concern in principle and could be fixed. Moreover, the carry tree appears to be functional.

Table 3 summarizes the characteristics of GATS.

Parameter	**Value**
Technology	AMS CUP 0.6μm 3M CMOS
Chip area	9.55mm^2
Devices (adder core)	12593
Devices (adder + pattern gen)	24033
Latency	2.3ns
Cycle time	1.0ns

Table 3. GATS statistics.

Chapter 5

Elliptic Curve Crypto Chip

WITH growing concerns about security in the IT landscape hardware support for cryptographic algorithms is becoming ever more important. To this end a fully pipelined Massey-Omura normal basis multiplier for use in elliptic curve cryptography (ECC) is presented in this chapter. We begin with some background in cryptography, elliptic curves and normal basis multiplication. Following, the chip architecture is described and the circuit building blocks are detailled. Finally, measurement results are presented and a summary is given.

5.1 Cryptography Background

The proliferation of the Internet in the last couple of years has resulted in a surge of the amount of sensitive electronic data being transferred. Possible applications include shopping via the web, electronic banking or secure email conversation. Cryptography is the field that deals with algorithms and applications to protect such secret data against eavesdroppers.

There are two important classes of cryptographic algorithms, i. e. symmetric and asymmetric algorithms. The former ones are characterized by a symmetric encryption and decryption process. One secret key serves both directions where decryption is done by stepwise reversing of encryption. Bulk encryption and decryption is performed by symmetric schemes like Triple-DES or AES. Throughput in the Giga range is possible.

The question is then how the secret key is shared between sender and receiver. That is where asymmetric schemes, known as public-key cryptosystems, come into play. Their distinguishing feature is that every party possesses a pair of private and public keys. The trick is that knowledge of the public key alone is of no use to the attacker whereas the secret private key is never sent down the wire.

Diffie and Hellman in [DH76] introduced a key exchange algorithm which allows sharing a secret key for symmetric encryption over an unsecure channel. It is based on the discrete logarithm problem which is believed to be computationally hard. Given a large finite group G and group elements $P_0, P \in G$ with $P = k \cdot P_0$ then according to current knowledge it is impossible to compute k given only P_0 and P. The reason for the name discrete *logarithm* stems from the fact that one notates the abstract group operation either additively, as before, or multiplicatively: ${P_0}^k = P$. In the latter notation k is the logarithm of P to the base P_0. The function $f(k, P) = k \cdot P$ is called a one-way or trapdoor function because f can be computed efficiently while f^{-1} cannot. Another example is the factoring problem which forms the basis of the RSA public key crypto system. It is currently deemed infeasible to factor a 1024-bit number $n = p \cdot q$ into their prime factors p and q.

Now suppose Alice wants to share a secret key with Bob. A large finite group G and a base point $P_0 \in G$ are fixed and public information. Alice picks an integer k_A as private key, and so does Bob with k_B. Alice then computes $P_A = k_A \cdot P_0$ whereas Bob computes $P_B = k_B \cdot P_0$. Now Alice and Bob exchange P_A and P_B and both compute the secret $P_s = k_A \cdot (k_B P_0) = k_B \cdot (k_A P_0)$. Note that Alice's and Bob's computations are easy while a potential attacker must solve the discrete logarithm for P_A and P_B to obtain the secret P_s.

5.1.1 Elliptic Curves

The difficulty of the discrete logarithm problem and thus the security of the key exchange depends not only on the order of the group G being used but also on the structure of G. The original proposal of Diffie and Hellman involved the multiplicative group F_p^* of the field of integers modulo a prime p. By using the point group of certain cubic equations the problem gets much more difficult. The motivation is to have higher security per bit as in the RSA scheme allowing one to use smaller keys. It is deemed that an elliptic curve key of 270 bits length provides at least the security of an RSA key with 2048 bits.

In the following we will only consider so-called non-supersingular elliptic curves as these provide the highest security. Non-supersingular elliptic curves are defined as the set of solutions of the cubic equation

$$y^2 + xy = x^3 + ax^2 + b \tag{5.1}$$

together with the point at infinity $\mathcal{O}$, where $a, b \in F, b \neq 0$ and F is any field. By defining an appropriate addition operation an elliptic curve becomes an abelian group. By varying the parameters a, b one has a source of finite abelian groups. For elliptic curves over the reals a geometric interpretation of the addition can be given: the points P and Q are added in that the third intersection point with the curve of a straight line through them is mirrored at the x axis.

For non-supersingular elliptic curves the basic operation of adding points $P = (x_1, y_1)$ and $Q = (x_2, y_2)$ is as follows:

$$R = P + Q = (x_3, y_3) \tag{5.2}$$

If $P \neq Q$, i.e. point addition:

$$\phi = \frac{y_2 - y_1}{x_2 - x_1} \tag{5.3}$$

$$x_3 = \phi^2 + \phi + x_1 + x_2 + a_2 \tag{5.4}$$

$$y_3 = \phi(x_1 + x_3) - y_1 + x_3 \tag{5.5}$$

If $P = Q = (x, y)$, i.e. point doubling:

$$\phi = x + \frac{y}{x} \tag{5.6}$$

$$x_3 = x^2 + \frac{b}{x^2} \tag{5.7}$$

$$y_3 = x^2 + (\phi + 1)x_3 \tag{5.8}$$

To avoid computing inverses, we change from affine to projective coordinates. If $P = (x_1, y_1, z_1)$ and $Q = (x_2, y_2, z_2)$ where $P, Q \neq \mathcal{O}$ and $P \neq -Q$ then $R = P + Q = (x_3, y_3, z_3)$ is for $P \neq Q$:

$$x_3 = AD \tag{5.9}$$

$$y_3 = CD + A^2(Bx_1 + Ay_1) \tag{5.10}$$

$$z_3 = A^3 z_1 z_2 \tag{5.11}$$

where $A = x_2z_1 + x_1z_2, B = y_2z_1 + y_1z_2, C = A + B$ and $D = A^2(A + az_1z_2) + z_1z_2BC$. We can fix z_2 to 1, as always $P_0 = (x_0, y_0, 1)$ will be added.

If $P = Q$, then

$$x_3 = AB \tag{5.12}$$

$$y_3 = x_1^4 A + B(x_1^2 + y_1z_1 + A) \tag{5.13}$$

$$z_3 = A^3 \tag{5.14}$$

where $A = x_1z_1, B = bz_1^4 + x_1^4$.

Thus, computing $P + Q$ requires 13 field multiplications and $2P$ requires 7 multiplications. A full double and add requires 20 multiplications. As doubling is cheaper in terms of multiplication as addition, the use of projective coordinates benefits from a k with low hamming weight.

The central task in an elliptic curve cryptosystem is the computation of the point multiplication $Q = k \cdot P_0 = \sum_{n=1}^{k} P_0$. The multiplication of an elliptic curve point P_0 by some $k > 1$ is performed as repeated double and add of the base point P_0 using the above equations for x_3, y_3, z_3. To this end addition and multiplication have to be done in the underlying field, in our case the Galois field $F_{2^{270}}$. A field of characteristic two is used for a hardware implementation as addition reduces then to simply XOR-ing corresponding bits.

5.1.2 Optimal Normal Basis Multiplication

The elements of $F_{2^{270}}$ are represented in the so-called Optimal Normal Basis (ONB). In the following multiplication in ONBs is explained.

Let α be an element of the field F_{2^m}, then the ONB can be formed as:

$$\{\alpha^{2^{m-1}}, \ldots, \alpha^{2^2}, \alpha^{2^1}, \alpha\}$$

Any element in F_{2^m} can be represented in normal basis form as :

$$A = a_{m-1}\alpha^{2^{m-1}} + \ldots + a_2\alpha^{2^2} + a_1\alpha^2 + a_0\alpha \tag{5.15}$$

or

$$A = \sum_{i=0}^{m-1} a_i \alpha^{2^i} \tag{5.16}$$

Squaring of a number amounts to just a left rotation of bits, as $(\alpha^{2^i})^2 = \alpha^{2^{i+1}}$ and $\alpha^{2^m} = \alpha$ by dint of Fermat's little theorem.

Multiplication of two numbers $A, B \in F_{2^m}$ proceeds as follows:

$$A = \sum_{i=0}^{m-1} a_i \alpha^{2^i} \tag{5.17}$$

$$B = \sum_{j=0}^{m-1} b_j \alpha^{2^j} \tag{5.18}$$

$$C = A \cdot B = \sum_{i=0}^{m-1} \sum_{j=0}^{m-1} a_i b_j \alpha^{2^i} \alpha^{2^j} = \sum_{k=0}^{m-1} c_k \alpha^{2^k} \tag{5.19}$$

where the last sum just says that the product is in the field as well and thus representable over the ONB. The same reasoning yields

$$\alpha^{2^i} \cdot \alpha^{2^j} = \sum_{k=0}^{m-1} \lambda_{ijk} \alpha^{2^k} \tag{5.20}$$

which when substituted into (5.19) gives

$$c_k = \sum_{i=0}^{m-1} \sum_{j=0}^{m-1} a_i b_j \lambda_{ijk}. \tag{5.21}$$

This can be modified to

$$c_k = \sum_{i=0}^{m-1} \sum_{j=0}^{m-1} a_{i+k} b_{j+k} \lambda_{ij0}, \tag{5.22}$$

requiring only the calculation of the λ_{ij0}'s. An ONB has the minimum number of $2m-1$ non-zero terms in λ_{ij0}. This is also the number of the wiring connections from the registers to the multiplier circuit. For details on how to fill the lambda matrix we refer to [Rosing98, section 4.5].

The coefficients c_k of the product $C = A \cdot B$ can finally be calculated as

$$c_k = a_k b_{k+1} \oplus \bigoplus_{i=1}^{m-1} a_{i+k}(b_{j_1(i)+k} \oplus b_{j_2(i)+k}). \tag{5.23}$$

where $k = 0, \ldots, m-1$. Referring to Table 4 containing the matrix for $m = 270$, the first terms look like this:

$$\begin{aligned} c_0 &= a_0 b_1 \oplus a_1(b_0 \oplus b_{104}) \oplus a_2(b_{104} \oplus b_{226}) \oplus a_3(b_{208} \oplus b_{258}) \oplus \ldots \\ c_1 &= a_1 b_2 \oplus a_2(b_1 \oplus b_{105}) \oplus a_3(b_{105} \oplus b_{227}) \oplus a_4(b_{209} \oplus b_{259}) \oplus \ldots \\ c_2 &= a_2 b_3 \oplus a_3(b_2 \oplus b_{106}) \oplus a_4(b_{106} \oplus b_{228}) \oplus a_5(b_{210} \oplus b_{260}) \oplus \ldots \\ &\vdots \end{aligned}$$

Therefore the ONB multiplier has rotating operand registers for A and B. The individual c_k coefficients are computed in a pipelined fashion. The circuit for c_k consists of a XOR compressor m bits wide with the inputs being the $a_{i+k}(b_{j_1(i)+k} \oplus b_{j_2(i)+k})$ terms in (5.23). The j_1 and j_2 index terms result in irregular wiring between the B operand register and the multiplier core. After one c_k computation has been launched into the pipeline, the operand registers A and B are left rotated and the computation of c_{k+1} is launched. The stream of result c_k bits is collected in a serial register for the product $C = A \cdot B$.

To quantify the benefit of using an AWP for normal basis multiplication let n be the latency of the circuit for c_k normalized to cycles. A non-pipelined sequential calculation of the product would then take a total latency of $m \cdot n$ cycles. In the pipelined case we have to fill the pipeline initially (having latency n); every of the $m-1$ subsequent c_k's takes only one cycle due to pipelining. The net result for the pipelining case is then $n + m - 1$ cycles. With $n = 3$, $m = 270$ in our case and 1ns cycle time the pipelined ONB multiplier has a total latency of 272ns compared to 810ns for a non-pipelined circuit.

i	j_1	j_2	i	j_1	j_2	i	j_1	j_2	i	j_1	j_2	i	j_1	j_2
000	1	—	054	112	240	108	82	246	162	138	244	216	58	186
001	0	104	055	77	94	109	35	174	163	75	81	217	91	98
002	104	226	056	156	266	110	115	171	164	27	189	218	209	228
003	208	258	057	185	255	111	119	239	165	121	153	219	199	237
004	60	99	058	23	216	112	54	152	166	167	168	220	13	16
005	129	160	059	98	132	113	28	77	167	122	166	221	37	194
006	89	196	060	4	259	114	170	192	168	154	166	222	21	88
007	127	180	061	9	160	115	36	110	169	122	191	223	101	130
008	41	159	062	19	65	116	14	239	170	29	114	224	206	268
009	61	260	063	50	87	117	12	145	171	110	175	225	103	257
010	19	52	064	18	195	118	230	238	172	119	231	226	2	105
011	71	144	065	62	161	119	111	172	173	34	263	227	25	208
012	15	117	066	50	244	120	152	263	174	83	109	228	92	218
013	220	238	067	202	250	121	165	190	175	30	171	229	140	237
014	37	116	068	43	135	122	167	169	176	147	231	230	118	146
015	12	72	069	48	149	123	29	154	177	47	253	231	172	176
016	200	220	070	86	143	124	84	265	178	136	204	232	34	253
017	194	249	071	11	20	125	95	242	179	96	126	233	184	247
018	51	64	072	15	38	126	137	179	180	7	90	234	79	157
019	10	62	073	200	211	127	7	197	181	41	187	235	74	212
020	71	87	074	80	235	128	159	213	182	107	134	236	139	198
021	38	222	075	139	163	129	5	100	183	203	246	237	219	229
022	101	215	076	27	93	130	89	223	184	233	254	238	13	118
023	58	256	077	55	113	131	97	206	185	57	157	239	111	116
024	132	207	078	156	192	132	24	59	186	40	216	240	54	145
025	106	227	079	234	248	133	106	259	187	91	181	241	94	141
026	92	188	080	74	201	134	42	182	188	26	107	242	85	125
027	76	164	081	163	245	135	68	203	189	82	164	243	49	137
028	113	153	082	108	189	136	48	178	190	121	264	244	66	162
029	123	170	083	174	264	137	126	243	191	155	169	245	81	202
030	84	175	084	30	124	138	162	197	192	78	114	246	108	183
031	142	147	085	142	242	139	75	236	193	36	248	247	35	233
032	45	150	086	49	70	140	93	229	194	17	221	248	79	193
033	252	262	087	20	63	141	146	241	195	64	88	249	17	201
034	173	232	088	195	222	142	31	85	196	6	161	250	51	67
035	109	247	089	6	130	143	70	150	197	127	138	251	43	261
036	115	193	090	97	180	144	11	53	198	213	236	252	33	46
037	14	221	091	187	217	145	117	240	199	210	219	253	177	232
038	21	72	092	26	228	146	141	230	200	16	73	254	184	204
039	211	215	093	76	140	147	31	176	201	80	249	255	57	267
040	158	186	094	55	241	148	45	47	202	67	245	256	23	102
041	8	181	095	125	266	149	44	69	203	135	183	257	207	225
042	134	260	096	179	205	150	32	143	204	178	254	258	3	105
043	68	251	097	90	131	151	53	262	205	96	267	259	60	133
044	46	149	098	59	217	152	112	120	206	131	224	260	9	42
045	32	148	099	4	209	153	28	165	207	24	257	261	52	251
046	44	252	100	129	214	154	123	168	208	3	227	262	33	151
047	148	177	101	22	223	155	191	265	209	99	218	263	120	173
048	69	136	102	256	268	156	56	78	210	199	214	264	83	190
049	86	243	103	225	269	157	185	234	211	39	73	265	124	155
050	63	66	104	1	2	158	40	212	212	158	235	266	56	95
051	18	250	105	226	258	159	8	128	213	128	198	267	205	255
052	10	261	106	25	133	160	5	61	214	100	210	268	102	224
053	144	151	107	182	188	161	65	196	215	22	39	269	103	269

Table 4. Optimal Normal Basis for $F_{2^{270}}$.

5.2 Chip Implementation

It has been mentioned in section 5.1.1 that elliptic curve cryptography is all about point multiplication $Q = k \cdot P_0$ which in turn relies on lots of finite field multiplications. The elliptic curve cryptography chip described here only realizes the finite field multiplication in F_{270} due to the sigificant effort incurred by full-custom AWPCMOS design. The Giga-Hertz finite field arithmetic is clearly the most challenging part of elliptic curve arithmetic. A full-blown point multiplication chip entails two further layers, i.e. the implementation of point doubling and adding according to the equations given in section 5.1.1 and the toplevel double-and-add algorithm that scans the scalar k. As the cycle times occurring in these three layers span six orders of magnitude — from Kilo-Hertz in the toplevel double-and-add to Giga-Hertz for the individual bits in the serial Massey-Omura multiplication — there exists a potential to save power by controlling the layers using asynchronous state machines. This is further detailled in [HKH00].

5.2.1 Overview

Figure 70 shows the functional units according to the die photo in Figure 71. The 270 bit wide register file on the left side holds the curve parameters a, b, the coordinates $X0, Y0$ of P_0, temporaries A, B, D — rf. equations (5.9) to (5.14) — and current coordinates X, Y, Z. The operands for multiplication are situated in special high speed registers to the right of the register file. The operands are connected to the multiplier core through irregular routing corresponding to the ONB. The multiplier logic consists of bitslices for AND's and XOR's together with a large XOR compressor that produces the result bits serially. The result bit stream is sampled into a register from which the multiplication result can then be read out in parallel. Register file, multiplication operands and result registers are connected via a 270 bit bus.

In order to obtain a square shape for the 270 bit datapath the register cells and multiplier slices were folded in two adjacent 135 bit structures. This has the further advantage that the closing wires for the rotating operand registers can be kept short thus minimzing parasitics.

All that is needed to control the multiplier is an exact 270 high speed pulses. These pulses have a cycle time of 1ns, i.e. the multiplier runs at a frequency of 1GHz and will retire one result bit every nanosecond. The pulses are generated by a LFSR-based counter and a pulse doubler.

For EBeam measurement a small version of the XOR compressor controlled by a VCO was put onto the chip as well.

5.2.2 Register File

For pin economy the wide registers are written and read serially through a single pad. This necessitates shifting capability of the register cell as shown in Figure 72. It is a simple master slave design where the two non-overlapping clock phases are directly supplied by input pads. It was deemed too dangerous to rely on the transition of a single clock input pad to be steep enough to allow for local clock inversion. During the point multiplication algorithm the registers are read and written in parallel by the bus.

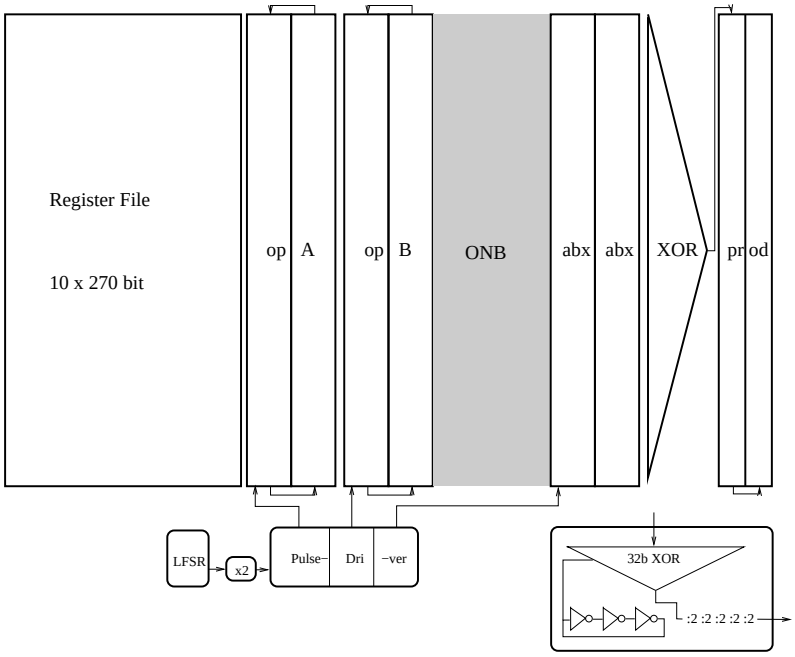


Figure 70. ECC chip overview.

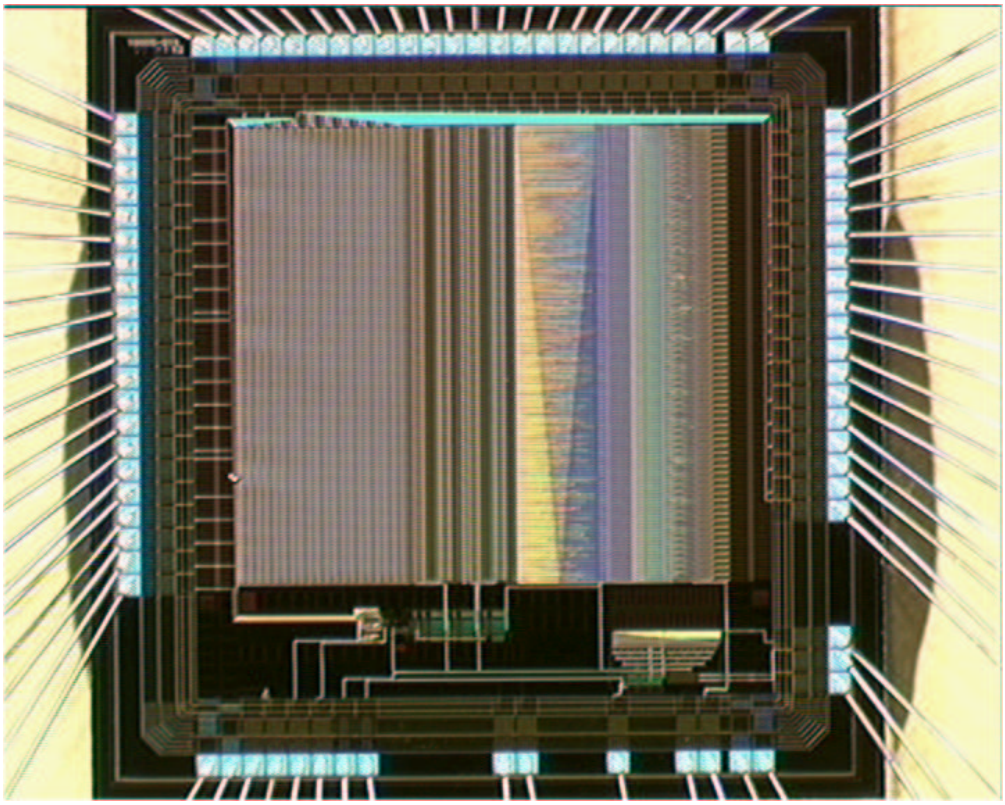


Figure 71. ECC die photo.

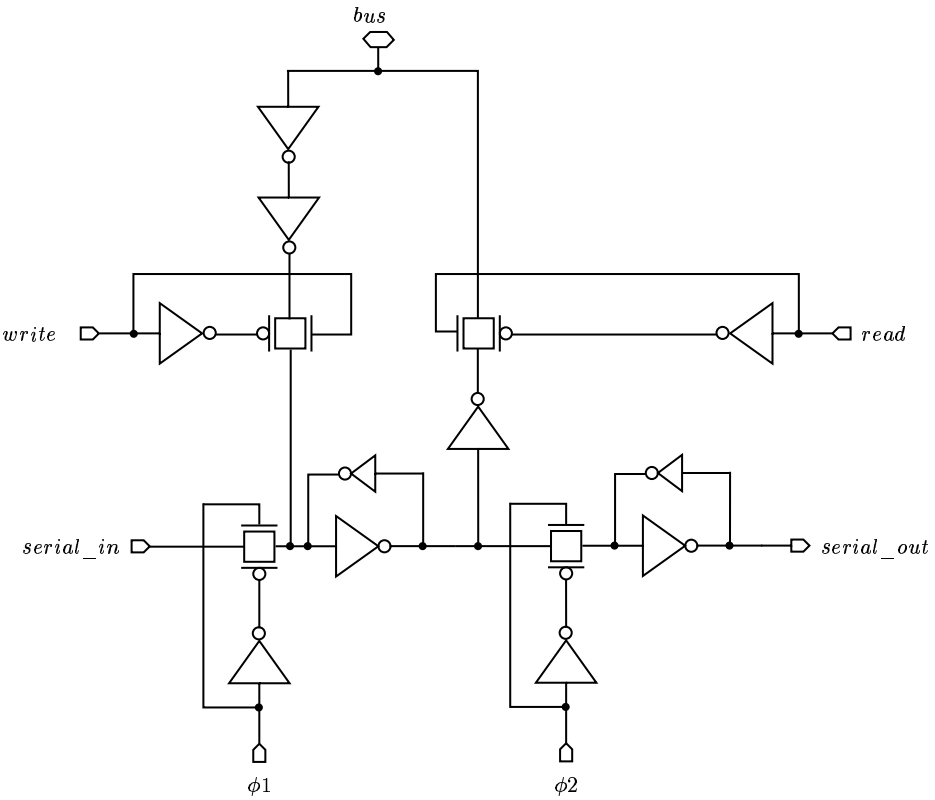


Figure 72. Basic shift cell of register file.

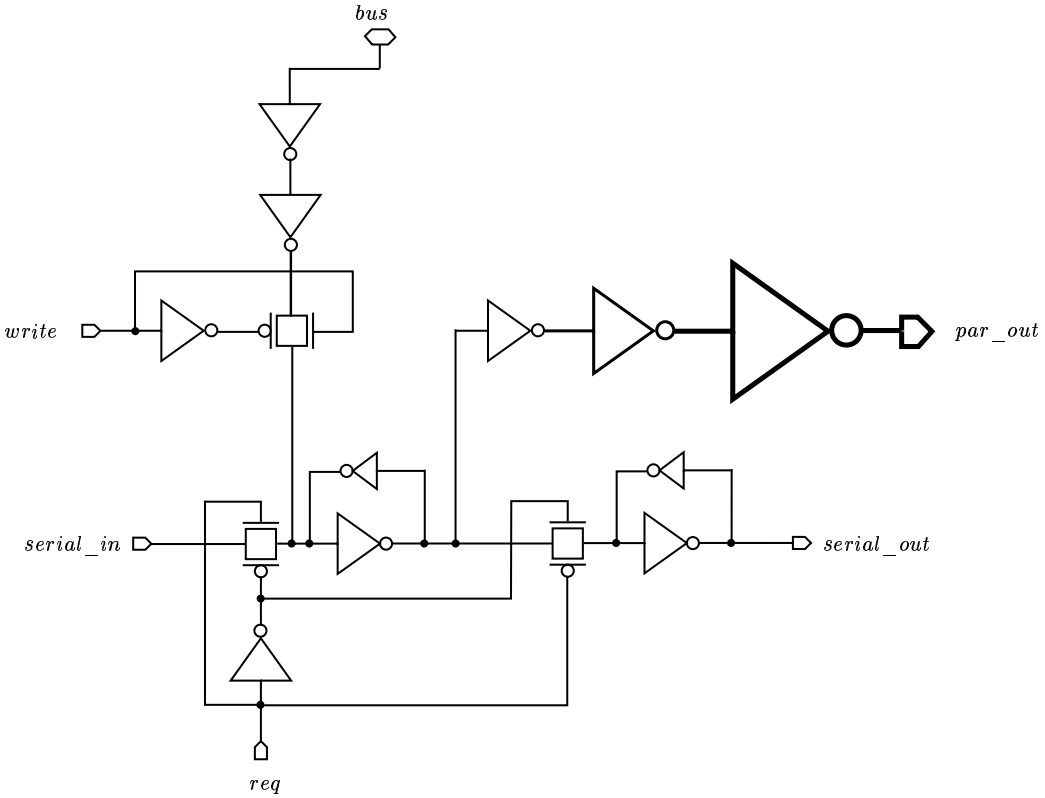


Figure 73. Basic shift cell of multiplier operand registers.

5.2.3 Operand Registers

A similar cell, rf. Figure 73, is also used for the write-only multiplicand registers, this time with local clock inversion to achieve high frequency. The contents is tapped after the master. Three tapered inverter buffers boost the drive in order to handle the ONB interconnect with parasitics in the picofarad range.

5.2.4 Massey-Omura Multiplier

The operands are leaving the rotating registers mentioned before as levels. This is advantageous for driving the big load represented by the ONB wiring. However, the multiplier core works with pulses so that a level to pulse conversion becomes necessary, rf. Figure 74.

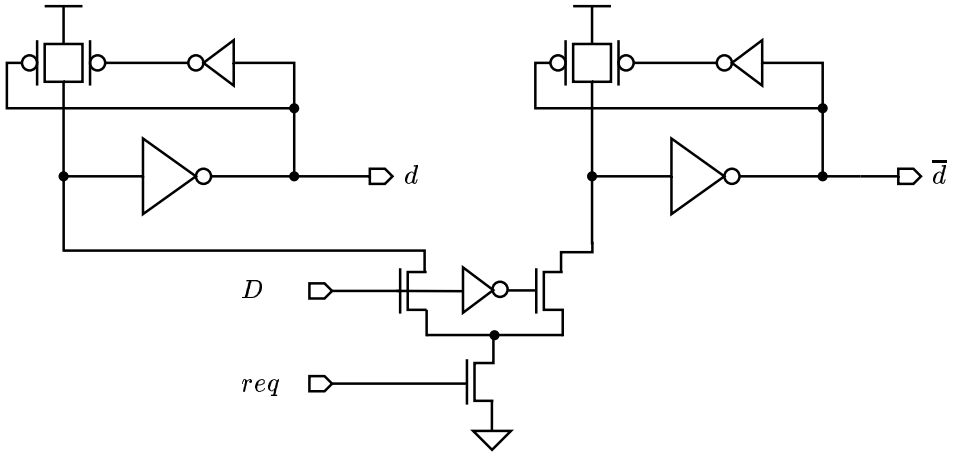


Figure 74. Dual-rail pulser circuit.

The request inputs of the dual-rail pulser in Figure 74 and the operands registers in Figure 73 are driven by the pulse counter. The rest of the circuit works asynchronously.

Following pulse conversion all summands of equation (5.23) are computed in parallel by the circuit shown in Figure 75. Two of these circuits together with their respective pulse converters and a XOR combining the two abx results are integrated in one bit slice. For the computation of the c_k in (5.23) it remains to compute a XOR of the 135 XOR outputs of the bit slices. This is accomplished by the XOR compressor tree shown in Figure 76.

The challenge in the layout of the XOR tree is to accurately balance interconnect whose lengths double while progressing through the stages of the tree from left to right.

5.2.5 Result Sampling Register

The purpose of the result register is to convert the serial bit stream originating from the XOR tree output into a parallel form. The product can then be written back into the register file and used further along in the algorithm. Unlike the operand registers, the contents of the product register is of interest only after all result bits have been collected. This gives a degree of freedom in the clocking of this register. A conventional shift register needs to be clocked globally with a load proportional to n, the number of entries in the shift register. In contrast, the sampling shift register used here to collect the product bits propagates the clock along the chain. Therefore the sampling shift register presents only a unit load for the clock. A simple pulse catcher circuit converts the dual-rail pulse from the XOR tree into level data and a local clock to feed the sampler.

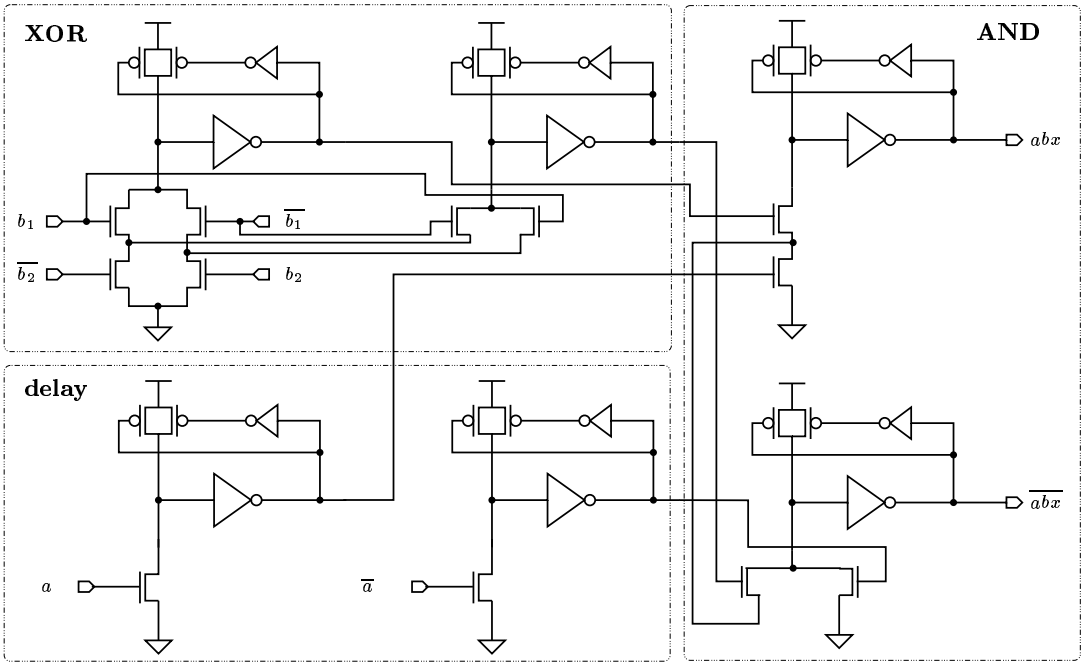


Figure 75. Circuit for $abx = a \cdot (b_1 \oplus b_2)$.

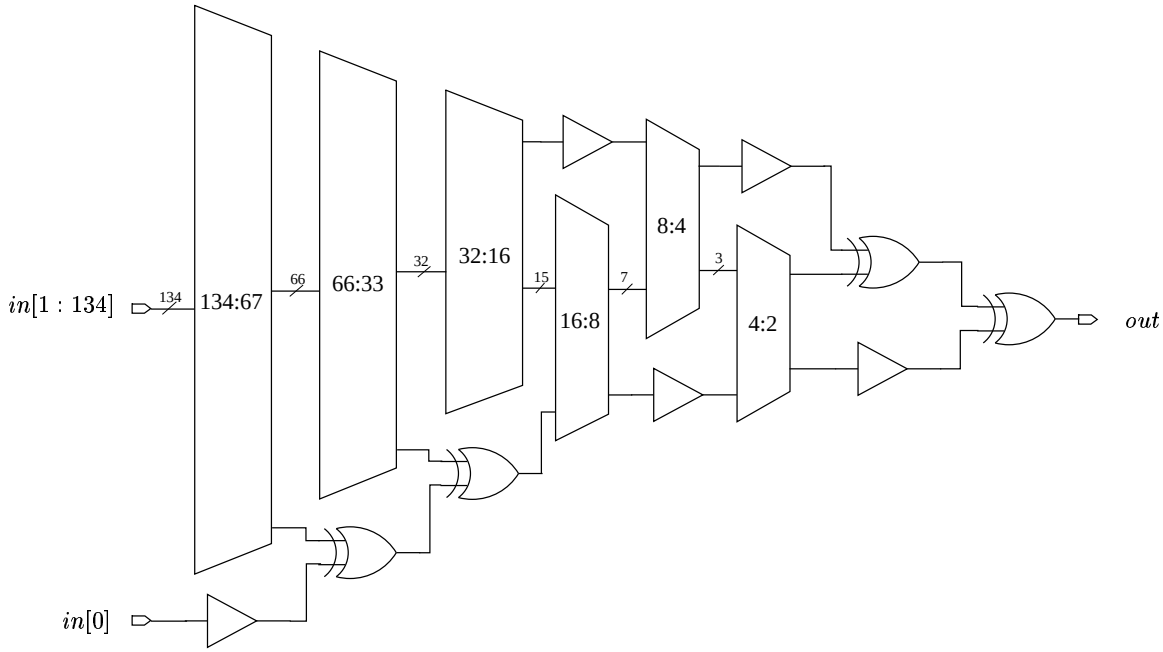


Figure 76. XOR compressor computing $out = \sum_{i=0}^{134} in_i = \bigoplus_{i=0}^{134} in_i$.

5.2.6 Pulse Counter

The multiplier needs an exact 270 pulses at 1ns cycle time. A conventional counter is ruled out as the carry logic wouldn't be able to complete in one cycle. Therefore a LFSR counter was chosen that doesn't care about intermediate states.[1] The counter increment path has only a shift and a XOR operation. The abort check consists of an OR over all entries which

[1]Note the analogy to the sampling shift register which outperforms a conventional shift register as it doesn't have to care about intermediate states either.

is fast when realized in AWPCMOS. But even this structure has difficulty meeting the 1ns cycle. Therefore the counter only generates 135 cycles which are sent to a pulse doubler whose operation is shown in Figure 77.

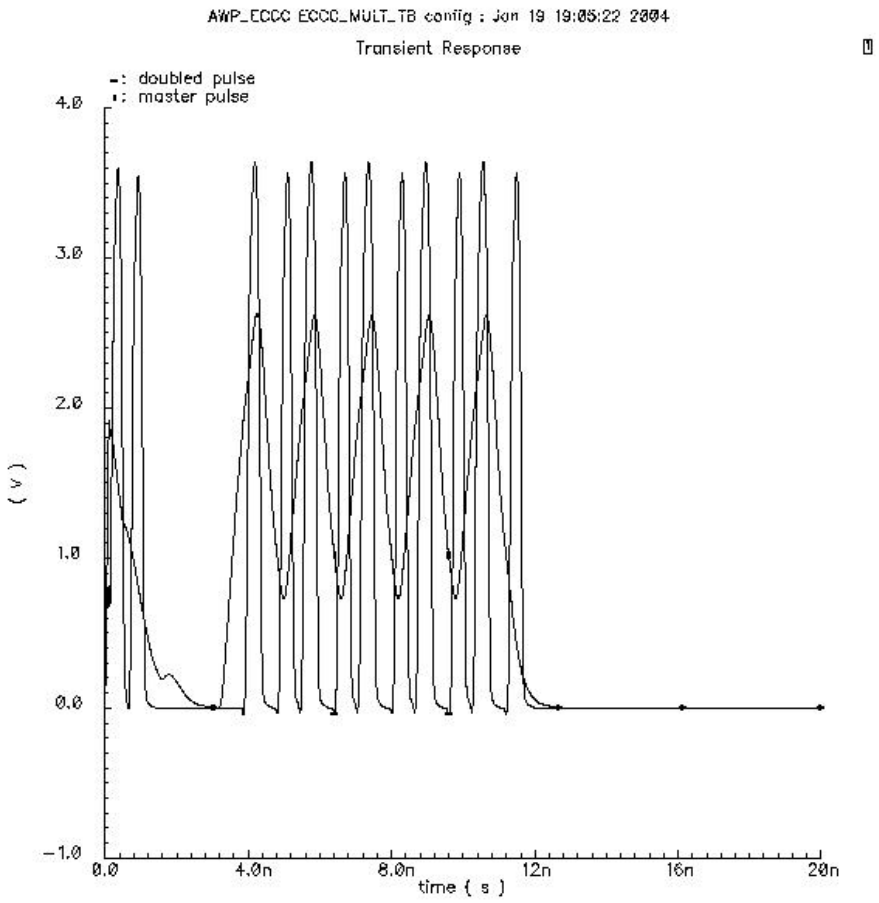


Figure 77. Master pulses (5) and doubled pulses (10).

5.2.7 AWP XOR Test Circuit

For testing purposes a 32 bit parity controlled by a VCO is included at the bottom right of the chip, rf. Figure 78. The XOR tree structure, the three big VCO inverters and the five divide-by-2 elements to the right are clearly visible. Both EBeam and via pad measurements are possible to explore AWPCMOS behaviour at varying frequencies, rf. section 5.3.

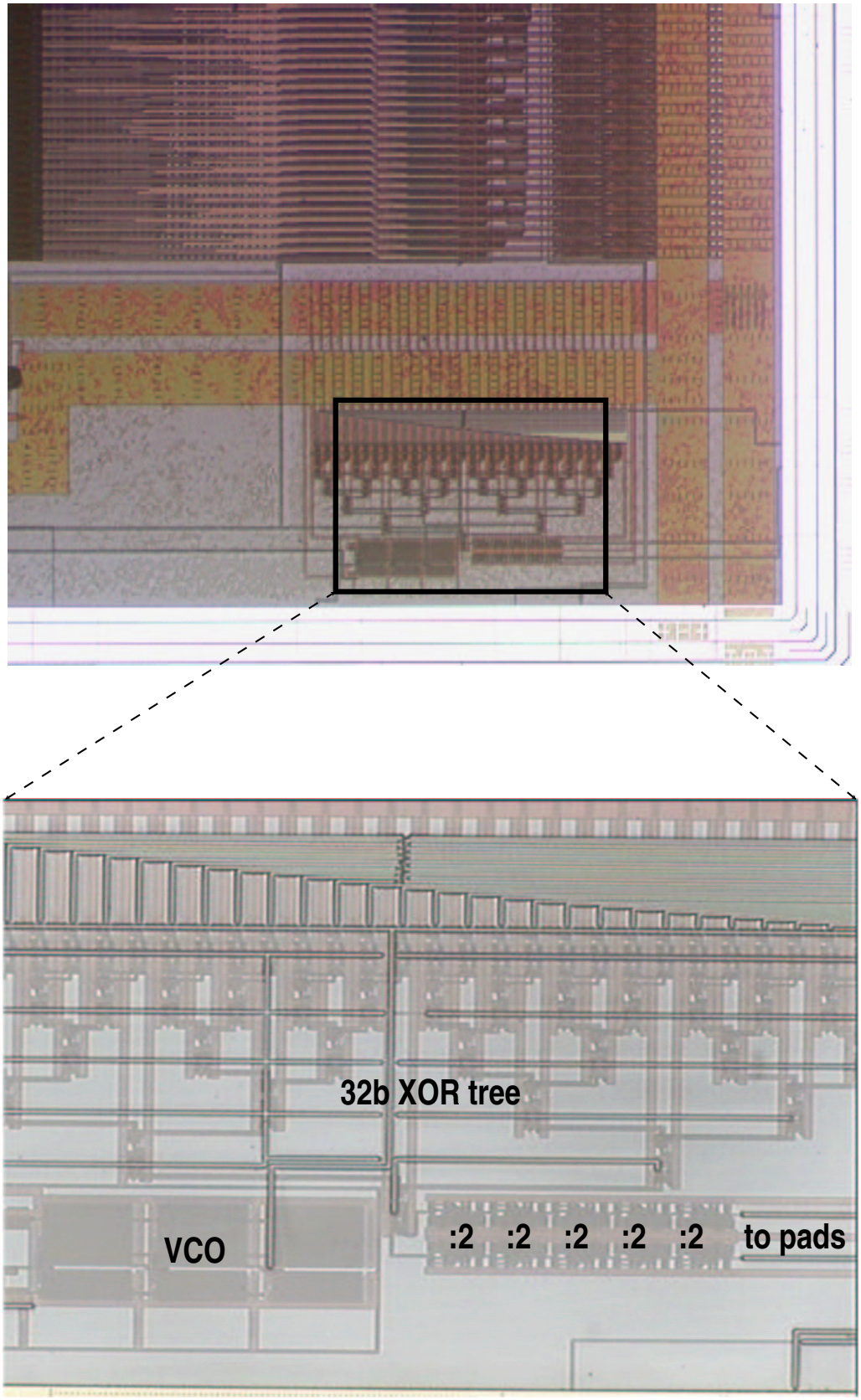


Figure 78. Closeup of AWP XOR test circuit.

5.3 Measurement Results

The ECC device was tested on a HP83000 tester. However the device behaved strangly so that an EBeam measurement was set up to investigate the root cause. The EBeamer is shown in Figure 79. After the DUT has been depassivated, it can measure periodic signals on upper metal layers with resolution up to 40ps without loading the circuit down.

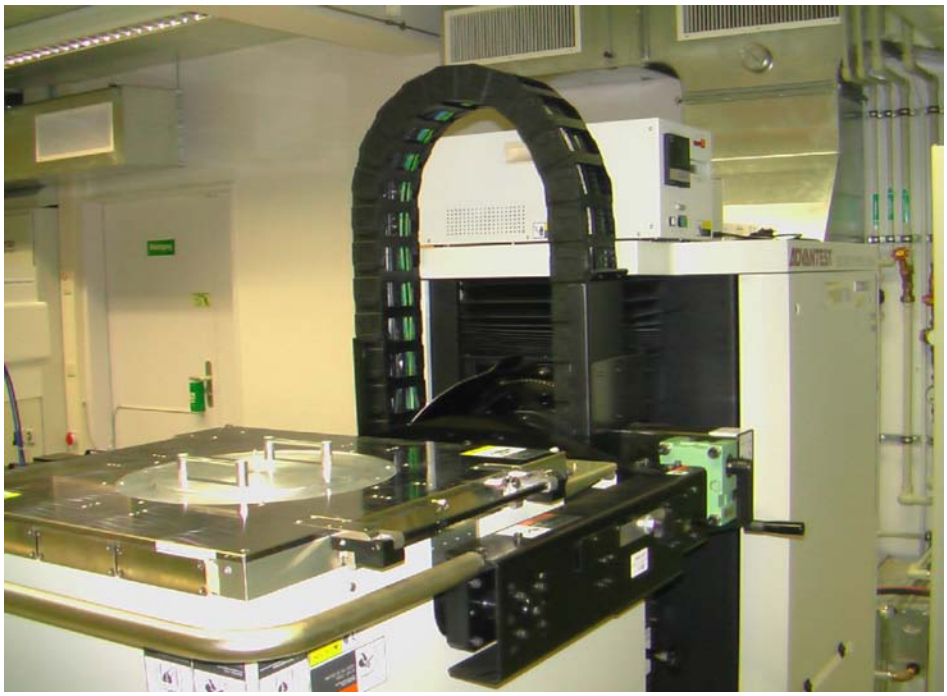


Figure 79. Advantest EBeam Test System with rotatable vaccum chuck.

For the actual measurement the DUT is mounted on the HP83000 tester head as shown in Figure 80. The EBeam head is rotated and placed on top of the tester head. A vacuum is then built up in the chamber between the tester and EBeamer heads where the DUT resides. This is because the electron gun located in the EBeamer head can only work in vacuum. Patterns are applied by the tester that causes the DUT to enter a short loop and triggering the EBeamer at every iteration. The electron ray is directed to the point of interest and the Ebeamer samples the voltage. Many samples are averaged to build the final trace.

Sadly, it turned out that the big slice drivers are failing unexpectedly thus rendering the datapath non functional. The top trace in Figure 81 corresponds to the double pumped pulse of Figure 77. This pulse is getting distorted in the preamplifiers and after the final amplifier there is no signal.

However the small XOR test circuit fully confirms the simulation, rf. Figure 82. The upper half shows on overlay of some signals belonging to the same sequential position in the tree. The jitter is only 30ps which is essential for the feasibility of AWPCMOS. The lower half shows 1GHz pulses propagating from the leaf nodes up to the root.

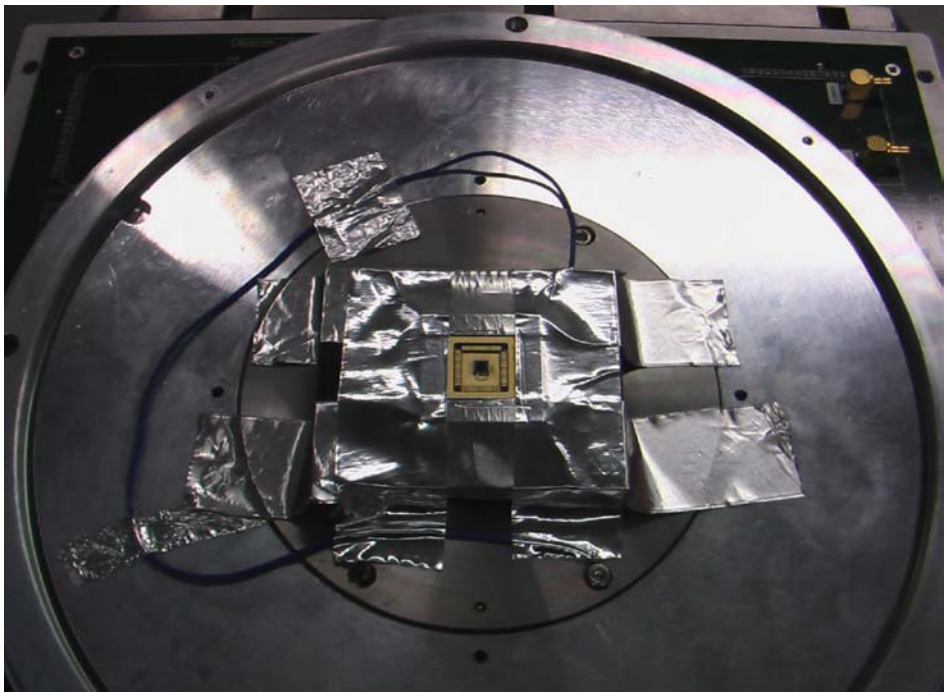


Figure 80. DUT mounted on HP83000 head for EBeam measurement. Grounded silver foil collects stray electrons.

Technology	AMS 0.35 μm 3M CMOS
Frequency	1.0 GHz max.
Core / IO voltage	3.3 V
Devices	$\approx$ 150000 full-custom
Area	$\approx 10\,\mathrm{mm}^2$
Package	CPGA 84

Table 5. ECC statistics.

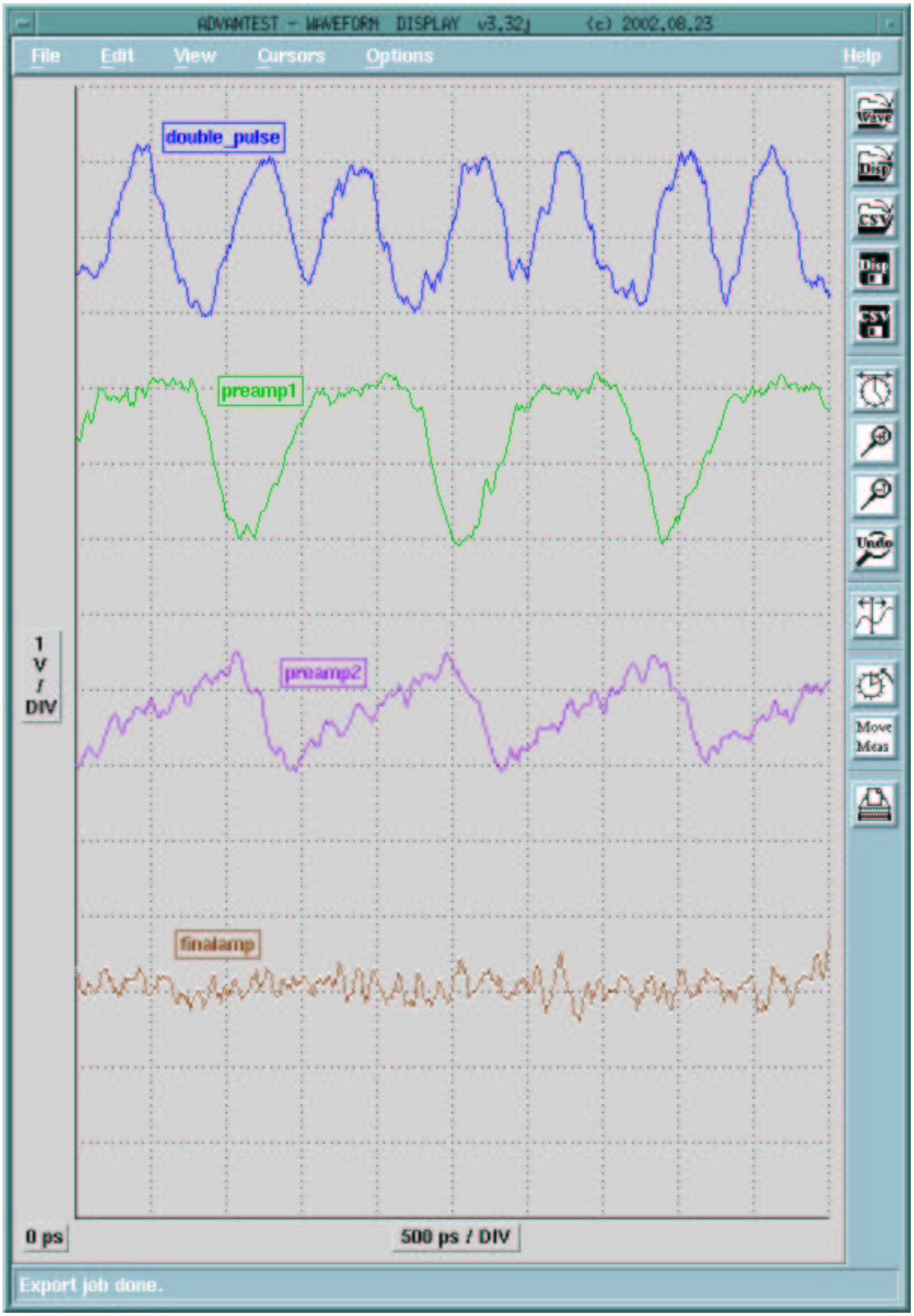
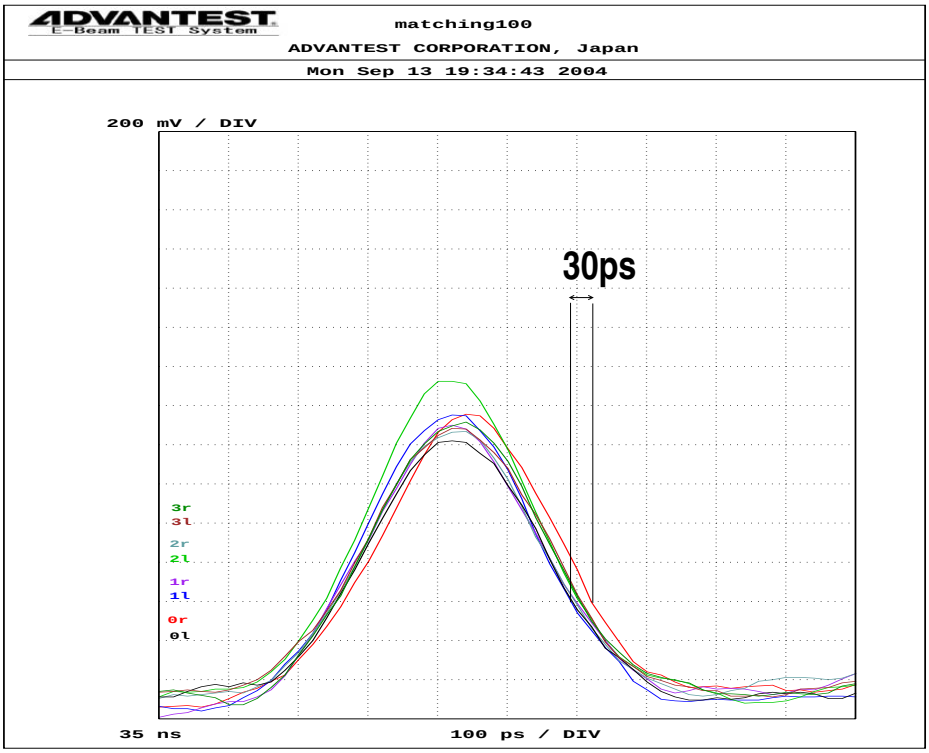


Figure 81. Dying pulses.



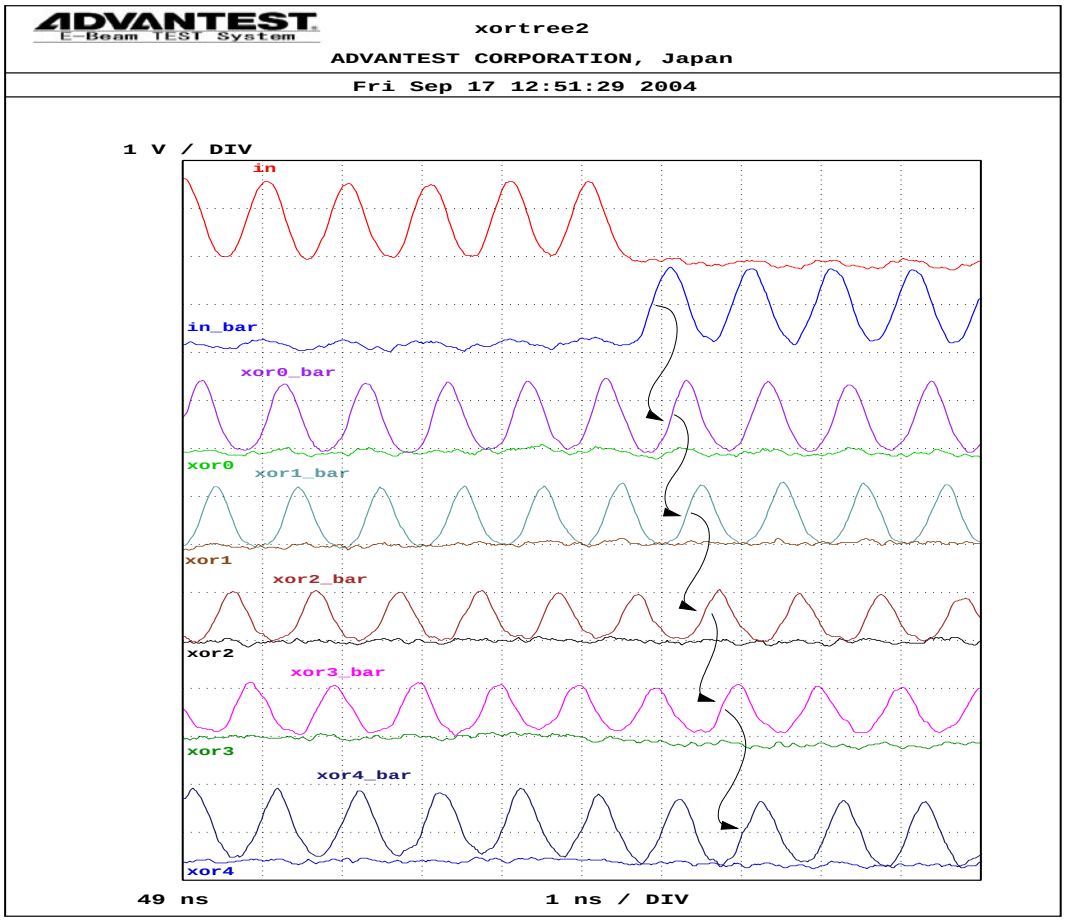


Figure 82. EBeam traces of XOR test circuit.

Chapter 6

Conclusion

6.1 Contributions

The work presented in this thesis contributes improvements to the wave pipelining method. It is shown that an asynchronous timing is more feasible than synchronous timing. This is due to two factors: first, only the asynchronous wave pipeline has a continuous frequency range; second, an asynchronous dual-rail pipeline does not need an explicit matched delay and is more robust than a wave pipeline with clocked synchronous input and output and intentional delays.

The AWPCMOS circuit style is shown with detailled SPICE simulations to have the best delay characteristics among various logic families proposed for wave pipelining. Static and pass gate logic has inherent drawbacks related to information encoded as levels. In contrast, AWPCMOS is a pulsed self-resetting logic having latency comparable to domino logic and a six inverter delay cycle. Even though AWPCMOS gates have no footer for maximum speed it is shown how cross current free operation can be achieved by correct sizing.

Monte Carlo simulations have been carried out in 0.35μm, 130nm and 90nm CMOS to assess the impact of process variations and device mismatch. It turns out, even in current 90nm technology, that AWPCMOS is well-behaved. These inter and intra die variations appear not to be a showstopper.

This work improves on [Klass94, Nowka95] as those only consider static logic styles and do not use Monte Carlo simulations.

It is worth noting that the AWP idea has been picked up by other researchers [SI01, PC00, LK00].

As far as the silicon experiments are concerned, the fabricated adder is the first 64-bit wave pipelined adder reported. The same is true for the Massey-Omura multiplier. The combination of 1.0GHz frequency and datapath width of 64 bits and 270 bits in 0.6μm and 0.35μm CMOS, respectively, represents the limit of these technologies. Simulations indicate that in 90nm CMOS 10GHz is feasible.

Unfortunately, both chips could not be fully characterized and functionally verified. In the case of the adder this is due to a timing problem in the final XOR, in the ECC chip a large driver fails for unknown reasons. However, the carry lookahead tree of the adder could be tested. The ECC chip integrates a test circuit containing a small version of the XOR compressor. EBeam measurements confirm analog simulations and demonstrate balanced delays in AWPCMOS.

6.2 Outlook

Even though the testchips in principle demonstrate the feasibility of the AWP concept it has not become entirely clear what is really the limiter for performance in practice: is it data dependency, process or noise ? This could be settled only with various test structures measured in silicon.

What has become clear during the design of the test vehicles is that the AWP sizing and tuning process could be well supported by an algorithm that iteratively extracts and tunes the layout to balance delays. Doing this manually and without the help of a Fastmos simulator as was done for this thesis is very tedious and time consuming.

When these low level issues have been settled it is worthwhile to look into system issues not covered here, like embedding AWPs in synchronous and asynchronous systems, feedback and testing.

Appendix A

List of Original Publications

1. O. Hauck, H. Sauerwein, and S. Huss, "Asynchronous VLSI Architectures for Huffman Codecs," *Proceedings IEEE International Symposium on Circuits and Systems*, Monterey, June 1998.

2. O. Hauck and S. Huss, "Asynchronous Wave Pipelines for High Throughput Datapaths," *Proceedings IEEE 5th International Conference on Electronics, Circuits and Systems*, pp. 283–286, Lisbon, September 1998.

3. O. Hauck, M. Garg, and S. Huss, "Efficient and Safe Asynchronous Wave-Pipeline Architectures for Datapath and Control Unit Applications," *Proceedings IEEE 9th Great Lakes Symposium on VLSI*, pp. 38–41, Ann Arbor, March 1999.

4. O. Hauck, M. Garg, and S. Huss, "Two-Phase Asynchronous Wave-Pipelines and their Application to a 2D-DCT," *Proceedings IEEE 5th International Symposium on Advanced Research in Asynchronous Circuits and Systems*, pp. 219–228, Barcelona, April 1999.

5. O. Hauck and S. Huss, "Asynchronous Wave-Pipelines with Improved Logic and Latch Circuits," *Proceedings IEEE 2nd Workshop on Design, Test, and Applications*, pp. 13–16, Dubrovnik, June 1999.

6. O. Hauck, S. Mietens, and S. Huss, "Giga-Hertz SRT-Division mit asynchronen Wave-Pipelines" *(in German)*, *9. E.I.S.-Workshop*, pp. 29–36, Darmstadt, September 1999.

7. O. Hauck and S. Huss, "Circuit Design for SRCMOS Asynchronous Wave Pipelines," *4th Asynchronous Circuit Design Workshop*, Grenoble, February 2000.

8. O. Hauck, A. Katoch, and S. Huss, "VLSI System Design Using Asynchronous Wave Pipelines: A 0.35μm CMOS 1.5GHz Elliptic Curve Public Key Cryptosystem Chip," *Proceedings IEEE 6th International Symposium on Advanced Research in Asynchronous Circuits and Systems*, pp. 188–197, Eilat, April 2000.

9. M. Ernst, S. Klupsch, O. Hauck, and S. Huss, "Rapid Prototyping for Hardware Accelerated Elliptic Curve Public-Key Cryptosystems," *Proceedings IEEE 12th International Workshop on Rapid System Prototyping*, Monterey, June 2001.

Appendix B

Supervised Diploma Theses

1. Helmut Sauerwein,
   Entwurf und Full-Custom Implementierung einer asynchronen VLSI-Architektur für Huffman-Codes,
   October 1997.

2. Stephan Mietens,
   VLSI-Entwurf eines asynchronen SRT-Ringdividierers,
   August 1998.

3. Manish Garg (IIT Bombay),
   Design and Implementation of an Asynchronously Wave-Pipelined 2D-DCT Processor,
   February 1999.

4. Atul Katoch, (IIT Bombay),
   Design and Implementation of an Asynchronous Wave Pipelined Elliptic Curve Crypto Chip,
   February 2000.

5. I. Jankowski,
   VLSI-Realisierung einer 0.3μm CMOS 1 GHz 1D-DCT mit asynchronen Wave-Pipelines,
   February 2001.

Bibliography

[AS92] M. Afghahi and C. Svensson, "Performance of Synchronous and Asynchronous Schemes for VLSI Systems," *IEEE Trans. on Computers*, vol. 41, pp. 858–872, July 1992.

[AMV93] G. Agnew, R. Mullin, and S. Vanstone, "An Implementation of Elliptic Curve Cryptosystems Over $F_{2^{155}}$," *IEEE J. on Selected Areas in Communications*, vol. 11, pp. 804–813, June 1993.

[Amrutur99] B. Amrutur, *Design and Analysis of Fast Low Power SRAMs*, PhD Thesis, Stanford University, Aug. 1999.

[Bailey01] D. Bailey, "Clock Distribution" chapter 13 in A. Chandrakasan, W. Bowhill, and F. Fox, eds., *Design Of High-Performance Microprocessor Circuits*, IEEE Press, 2001.

[BCD+98] K. Bernstein, K. Carrig, C. Durham *et al.*, *High Speed CMOS Design Styles*, Kluwer Academic Publishers, 1998.

[BG97] V. Bartlett and E. Grass, "Completion-detection technique for dynamic logic," *Electronics Letters*, vol. 33, pp. 1850–1852, Oct. 1997.

[BJM+05] W. Belluomini, D. Jamsek, A. Martin *et al.*, "An 8Ghz Floating-Point Multiply," *ISSCC Digest of Technical Papers*, pp. 374–375, Feb. 2005.

[BK82] R. Brent and H. Kung, "A regular layout for parallel adders," *IEEE Trans. on Computers*, vol. C-31, pp. 260–264, Mar. 1982.

[BKYK99] P. Beerel, S. Kim, P.-C. Yeh, and K. Kim, "Statistically Optimized Asynchronous Barrel Shifters for Variable Length Codecs," *Proceedings Intern. Symp. Low Power Electronics and Design*, pp. 261–263, 1999.

[BNW98] M. Benes, S. Nowick, and A. Wolfe, "A Fast Asynchronous Huffman Decoder for Compressed-Code Embedded Processors," *Proceedings IEEE Intern. Symp. on Advanced Research in Asynchronous Circuits and Systems*, Apr. 1998.

[Bryant86] R. Bryant, "Graph-Based Algorithms for Boolean Function Manipulation," *IEEE Trans. on Computers*, pp. 677–691, Aug. 1986.

[BCK+98] W. Burleson, M. Ciesielski, F. Klass *et al.*, "Wave-Pipelining: A Tutorial and Research Survey," *IEEE Trans. on VLSI*, vol. 6, pp. 464–474, Sep. 1998.

[CSR05] S. Chan, K. Shepard, and P. Restle, "Uniform-Phase Uniform-Amplitude Resonant-Load Global Clock Distributions," *IEEE J. of Solid-State Circuits*, vol. 40, pp. 102–109, Jan. 2005.

[CCS+91] T. Chappell, B. Chappell, S. Schuster *et al.*, "A 2-ns Cycle, 3.8-ns Access 512-kb CMOS ECL SRAM with a Fully Pipelined Architecture," *IEEE J. of Solid-State Circuits*, vol. 26, pp. 1577–1585, Nov. 1991.

[Cheng98] F.-C. Cheng, "Practical Design and Performance Evaluation of Completion Detection Circuits," *Proceedings Intern. Conf. Computer Design*, pp. 354–359, 1998.

[CHM+01] L. Clark, E. Hoffman, J. Miller *et al.*, "An Embedded 32-b Microprocessor Core for Low-Power and High-Performance Applications," *IEEE J. of Solid-State Circuits*, vol. 36, pp. 1599–1608, Nov. 2001.

[CLJ+01] W. Coates, J. Lexau, I. Jones *et al.*, "FLEETzero: An Asynchronous Switching Experiment," *Proceedings IEEE Intern. Symp. on Advanced Research in Asynchronous Circuits and Systems*, pp. 173–182, 2001.

[Cotten69] L. Cotten, "Maximum-rate pipeline systems," *1969 AFIPS Proc. Spring Joint Computer Conf.*, vol. 34, pp. 581–586, Montvale, NJ: AFIPS Press, May 1969.

[DBG+05] D. Deleganes, M. Barany, G. Geannopoulos *et al.*, "Low-Voltage Swing Logic Circuits for a Pentium 4 Processor Integer Core," *IEEE J. of Solid-State Circuits*, vol. 40, pp. 36–43, Jan. 2005.

[DH76] W. Diffie and M. Hellman, "New directions in cryptography," *IEEE Trans. on Inform. Theory*, vol. 22, pp. 644–654, 1976.

[DW95] P. Day and J. Woods, "Investigation into Micropipeline Latch Design Styles," *IEEE Trans. on VLSI*, vol. 3, pp. 264–272, Jun. 1995.

[Ebergen01] J. Ebergen, "Squaring the FIFO in GasP," *Proceedings IEEE Intern. Symp. on Advanced Research in Asynchronous Circuits and Systems*, pp. 194–205, 2001.

[FS95] G. Fernandez and R. Sridhar, "Dual Rail Static CMOS Architecture for Wave Pipelining," *Proceedings 9th Intern. Conf. on VLSI Design*, pp. 335-336, Jan. 1996.

[GBP+98a] H. van Gageldonk, K. van Berkel, A. Peeters *et al.*, "An Asynchronous Low-Power 80C51 Microcontroller," *Proceedings IEEE Intern. Symp. on Advanced Research in Asynchronous Circuits and Systems*, pp. 96–107, 1998.

[GN95] D. Ghosh and S. Nandy, "Design and Realization of High-Performance Wave-Pipelined 8 x 8 b Multiplier in CMOS Technology," *IEEE Trans. on VLSI*, vol. 3, pp. 36–48, Mar. 1995.

[GBK97] E. Grass, V. Bartlett, and I. Kale, "Completion Detection Techniques for Asynchronous Circuits," *IEICE Trans. Inf. & Syst.*, vol. E80-D, pp. 344–350, March 1997.

[GLC94a] C. Gray, W. Liu, and R. Cavin, *Wave Pipelining: Theory and CMOS Implementation*, Kluwer, 1994.

[GLC94b] C. Gray, W. Liu, and R. Cavin, "Timing Constraints for Wave-Pipelined Systems," *IEEE Trans. on CAD*, vol. 13, pp. 987–1004, Aug. 1994.

[GBP+98b] P. Gronowski, W. Bowhill, R. Preston *et al.*, "High-Performance Microprocessor Design," *IEEE J. of Solid-State Circuits*, vol. 33, pp. 676–686, May 1998.

[Harris99] D. Harris, *Skew-Tolerant Circuit Design*, PhD Thesis, Stanford University, Feb. 1999.

[HH97] D. Harris and M. Horowitz, "Skew-Tolerant Domino Circuits," *ISSCC Digest of Technical Papers*, pp. 422–423, 1997.

[HH98] O. Hauck and S. Huss, "Asynchronous Wave Pipelines for High Throughput Datapaths," *Proceedings 5th IEEE Intern. Conf. on Electronics, Circuits and Systems*, pp. 283–286, Sep. 1998.

[HKH00] O. Hauck, A. Katoch, and S. Huss, "VLSI System Design Using Asynchronous Wave Pipelines: A 0.35μm CMOS 1.5GHz Elliptic Curve Public Key Cryptosystem Chip," *Proceedings IEEE Intern. Symp. on Advanced Research in Asynchronous Circuits and Systems*, pp. 188–197, Eilat, Apr. 2000.

[HSH98] O. Hauck, H. Sauerwein, and S. Huss, "Asynchronous VLSI Architectures for Huffman Codecs," *Proceedings IEEE Intern. Symp. on Circuits and Systems*, June 1998.

[HAA+00] R. Heald, K. Aingaran, C. Amir *et al.*, "A Third-Generation SPARC V9 64-b Microprocessor," *IEEE J. of Solid-State Circuits*, vol. 35, pp. 1526–1538, Nov. 2000.

[HSR+98] R. Heald, K. Shin, V. Reddy *et al.*, "64-KByte Sum-Addressed-Memory Cache with 1.6-ns Cycle and 2.6-ns Latency," *IEEE J. of Solid-State Circuits*, vol. 33, pp. 1682–1689, Nov. 1998.

[HUS+01] G. Hinton, M. Upton, D. Sager *et al.*, "A 0.18-μm CMOS IA-32 Processor With a 4-GHz Integer Execution Unit," *IEEE J. of Solid-State Circuits*, vol. 36, pp. 1617–1627, Nov. 2001.

[H+02] M. S. Hrishikesh *et al.*, "The Optimal Logic Depth Per Pipeline Stage Is 6 to 8 FO4 Inverter Delays," *Proc. 29th Int'l Symp. Computer Architecture,* pp. 14–24, 2002.

[HGS+99] W. Hwang, G. Gristede, P. Sanda, S. Wang, and D. Heidel, "Implementation of a Self-Resetting CMOS 64-Bit Parallel Adder with Enhanced Testability," *IEEE J. of Solid-State Circuits*, vol. 34, pp. 1108–1117, Aug. 1999.

[IEEE99] Special Issue of the *Proceedings of the IEEE* on Asynchronous Design, Feb. 1999.

[Kinniment96] D. Kinniment, "An Evaluation of Asynchronous Addition," *IEEE Trans. on VLSI*, vol. 4, pp. 137–140, March 1996.

[Klass94] E. Klass, *Wave Pipelining: Theoretical and Practical Issues in CMOS*, PhD Thesis, Delft University, Sep. 1994.

[KS73] P. Kogge and H. Stone, "A parallel algorithm for the efficient solution of a general class of recurrence equations," *IEEE Trans. on Computers*, vol. C-22, pp. 786–793, Aug. 1973.

[KTT+88] S. Komori, H. Takata, T. Tamura *et al.*, "An Elastic Pipeline Mechanism by Self-Timed Circuits," *IEEE J. of Solid-State Circuits*, vol. 23, pp. 111–117, Feb. 1988.

[LJK+05] S.-B. Lee, S.-J. Jang, J.-S. Kwak *et al.*, "A 1.6-Gb/s/pin Double Data Rate SDRAM With Wave-Pipelined CAS Latency Control," *IEEE J. of Solid-State Circuits*, vol. 40, pp. 223–232, Jan. 2005.

[LB95] W. Lien and W. Burleson, "Wave-Domino Logic: Theory and Applications," *IEEE Trans. on Circuits and Systems II*, vol. 42, pp. 78–91, Feb. 1995.

[LK00] B.-H. Lim and J.-K. Kang, "A Self-Timed Wave Pipelined Adder Using Data Align Method," *Proceedings 2nd IEEE Asia Pacific Conference on ASICs*, pp. 77–80, Aug. 2000.

[LGF+94] W. Liu, C. T. Gray, D. Fan *et al.*, "A 250-MHz Wave Pipelined Adder in 2-μm CMOS," *IEEE J. of Solid-State Circuits*, vol. 29, pp. 1117–1128, Sep. 1994.

[MMA+98] K. Mai, T. Mori, B. Amrutur *et al.*, "Low-Power SRAM Design Using Half-Swing Pulse-Mode Techniques," *IEEE J. of Solid-State Circuits*, vol. 33, pp. 1659–1671, Nov. 1998.

[MMN+95] H. Morinaka, H. Makino, Y. Nakase *et al.*, "A 64bit Carry Look-ahead CMOS Adder using Modified Carry Select," *Proceedings Custom Integr. Circ. Conf.*, pp. 585–588, 1995.

[MS97] P. Markovic and D. Simic, "A Combination of Wave-Pipelining Timing Methodology and DPTL Circuit Design Technique," *Proceedings 21st Intern. Conf. on Microelectronics*, pp. 787–790, Sep. 1997.

[MYH+03] F. O'Mahony, C. Yue, M. Horowitz *et al.*, "A 10-GHz Global Clock Distribution Using Coupled Standing-Wave Oscillators," *IEEE J. of Solid-State Circuits*, vol. 38, pp. 1813–1820, Nov. 2003.

[Naffziger96] S. Naffziger, "A sub-nanosecond 0.5μm 64b adder design," *ISSCC Digest of Technical Papers*, pp. 362–363, Feb. 1996.

[NGS05] S. Naffziger, T. Grutkowski, and B. Stackhouse, "The Implementation of a 2-core Multi-Threaded Itanium Family Processor," *ISSCC Digest of Technical Papers*, Feb. 2005.

[NYB97] S. Nowick, K. Yun, and P. Beerel, "Speculative Completion for the Design of High-Performance Asynchronous Dynamic Adders," *Proceedings IEEE Intern. Symp. on Advanced Research in Asynchronous Circuits and Systems*, Eindhoven, Apr. 1997.

[NH97] K. Nowka and P. Hofstee, "Circuits and Microarchitecture for Gigahertz VLSI Designs," *Proceedings 17th Conference on Advanced Research in VLSI*, Sep. 1997.

[Nowka95] K. Nowka, *High-Performance CMOS System Design Using Wave Pipelining*, PhD Thesis, Stanford University, Aug. 1995.

[PBS+96] H. Partovi, R. Burd, U. Salim *et al.*, "Flow-Through Latch and Edge-Triggered Flip-flop Hybrid Elements," *ISSCC Digest of Technical Papers*, pp. 138–139, 1996.

[PC00] C. Park and D. Chung, "Modified asynchronous wave-pipelining," *Electronics Letters*, vol. 36, pp. 295–297, Feb. 2000.

[PS97] R. Parthasarathy and R. Sridhar, "Double Pass Transistor Logic for High Performance Wave Pipeline Circuits," *Proc. Intern. Conf. on VLSI*, pp. 495–500, 1997.

[RCE+02] P. Restle, C. Carter, J. Eckhardt *et al.*, "The Clock Distribution of the Power4 Microprocessor," *ISSCC Digest of Technical Papers*, Feb. 2002.

[RMW+01] P. Restle, T. McNamara, D. Webber *et al.*, "A Clock Distribution Network for Microprocessors," *IEEE J. of Solid-State Circuits*, vol. 36, pp. 792–799, May 2001.

[Rogenmoser96] R. Rogenmoser, *The Design of High-Speed Dynamic CMOS Circuits for VLSI*, PhD Thesis, ETH Zurich, 1996.

[Rosing98] M. Rosing, *Implementing Elliptic Curve Cryptography*, Manning, 1998.

[Ruiz98] G. Ruiz, "Evaluation of Three 32-Bit CMOS Adders in DCVS Logic for Self-Timed Circuits," *IEEE J. of Solid-State Circuits*, vol. 33, pp. 604–613, Apr. 1998.

[SCC+92] S. Schuster, T. Chappell, B. Chappell, and R. Franch, "On-Chip Test Circuitry for a 2-ns Cycle, 512-kb CMOS ECL SRAM," *IEEE J. of Solid-State Circuits*, vol. 27, pp. 1073–1079, July 1992.

[Seitz80] C. Seitz, "System Timing" chapter 7 in C. Mead and L. Conway, *Introduction to VLSI Systems*, Addison-Wesley, 1980.

[SF01] I. Sutherland and S. Fairbanks, "GasP: A Minimal FIFO Control," *Proceedings IEEE Intern. Symp. on Advanced Research in Asynchronous Circuits and Systems*, pp. 46–53, Mar. 2001.

[SF01] J. Sparso and S. Furber, eds., *Principles of Asynchronous Circuit Design*, Kluwer Academic Publishers, 2001.

[SI01] T. Säntti and J. Isoaho, "Modified SRCMOS Cell for High-Throughput Wave-Pipelined Arithmetic Units," *Proceedings IEEE Intern. Symp. on Circuits and Systems*, vol. IV, pp. 194–197, 2001.

[SL01] I. Sutherland and J. Lexau, "Designing Fast Asynchronous Circuits," *Proceedings IEEE Intern. Symp. on Advanced Research in Asynchronous Circuits and Systems*, 2001.

[SM00a] M. Singh and S. Nowick, "High-Throughput Asynchronous Pipelines for Fine-Grain Dynamic Datapaths," *Proceedings IEEE Intern. Symp. on Advanced Research in Asynchronous Circuits and Systems*, pp. 198–209, Eilat, Apr. 2000.

[SMM00b] M. Singh and S. Nowick, "Fine-Grain Pipelined Asynchronous Adders for High-Speed DSP Applications," *Proc. IEEE Computer Society Annual Workshop on VLSI*, Apr. 2000.

[SMM01] M. Singh and S. Nowick, "MOUSETRAP: Ultra-High-Speed Transition-Signaling Asynchronous Pipelines," *Proceedings Intern. Conf. Computer Design*, pp. 9–17, 2001.

[STR+02] M. Singh, J. Tierno, A. Rylyakov *et al.*, "An Adaptively-Pipelined Mixed Synchronous-Asynchronous Digital FIR Filter Chip Operating at 1.3 Giga-Hertz," *Proceedings IEEE Intern. Symp. on Advanced Research in Asynchronous Circuits and Systems*, 2002.

[SMM96] R. Sridhar, A. Martinez-Smith, and B. McGee, "Speed and Power Comparison of CMOS Wave Pipelined Systems and Low Power WTGL," *Proceedings IEEE Intern. Symp. on Circuits and Systems*, pp. 156–159, 1996.

[SRG+01] K. Stevens, S. Rotem, R. Ginosar *et al.*, "An Asynchronous Instruction Length Decoder," *IEEE J. of Solid-State Circuits*, vol. 36, pp. 217–228, Feb. 2001.

[Sutherland89] I. Sutherland, "Micropipelines," *Communications of the ACM*, vol. 32, pp. 720–738, June 1989.

[S+93] M. Suzuki *et al.*, "A 1.5ns 32-b CMOS ALU in Double Pass-Transistor Logic," *IEEE J. of Solid-State Circuits*, vol. 28, pp. 1145–1151, Nov. 1993.

[S+98] J. Silberman *et al.*, "A 1.0-GHz Single-Issue 64-Bit PowerPC Integer Processor," *IEEE J. of Solid-State Circuits*, vol. 33, pp. 1600–1608, Nov. 1998.

[SZ98] R. Sridhar and X. Zhang, "Method and Apparatus for Designing Circuits for Wave Pipelining," *U.S. Patent #5,796,624*, Aug. 1998.

[THT+95] S. Tachibana, H. Higuchi, K. Takasugi *et al.*, "A 2.6-ns Wave-Pipelined CMOS SRAM with Dual-Sensing-Latch Circuits," *IEEE J. of Solid-State Circuits*, vol. 30, pp. 487–490, Apr. 1995.

[TCB+00] T.-Y. Tang, C.-S. Choy, J. Butas, and C.-F. Chan, "An ALU Design using a Novel Asynchronous Pipeline Architecture," *Proceedings IEEE Intern. Symp. on Circuits and Systems*, vol. V, pp. 361–364, May 2000.

[WYS00] J.-S. Wang, P.-H. Yang, and D. Sheng, "Design of a 3-V 300-MHz Low-Power 8-b x 8-b Pipelined Multiplier Using Pulse-Triggered TSPC Flip-Flops," *IEEE J. of Solid-State Circuits*, vol. 35, pp. 583–592, Apr. 2000.

[WH91] T. Williams and M. Horowitz, "A Zero-Overhead Self-Timed 160-ns 54-b CMOS Divider," *IEEE J. of Solid-State Circuits*, vol. 26, pp. 1651–1661, Nov. 1991.

[Williams01] T. Williams, "Self-Timed Pipelines" chapter 9 in A. Chandrakasan, W. Bowhill, and F. Fox, eds., *Design Of High-Performance Microprocessor Circuits*, IEEE Press, 2001.

[Woo88] A. Woo, "Static PLA or ROM Circuit with Self-Generated Precharge," *U.S. Patent #4,728,827*, Mar. 1988.

[WWM+05] J. Wuu, D. Weiss, C. Morganti, and M. Dreesen, "The Asynchronous 24MB On-Chip Level-3 Cache for a Dual-Core Itanium-Family Processor," *ISSCC Digest of Technical Papers*, Feb. 2005.

[YCC01] J.-L. Yang, C.-S. Choy, and C.-F. Chan, "A Self-Timed Divider Using a New Fast and Robust Pipeline Scheme," *IEEE J. of Solid-State Circuits*, vol. 36, pp. 917–923, June 2001.

[Y+90] K. Yano *et al.*, "A 3.8ns CMOS 16x16 Multiplier Using Complementary Pass Transistor Logic," *IEEE J. of Solid-State Circuits*, vol. 25, pp. 388–395, April 1990.

[YBA96] K. Yun, P. Beerel, and J. Arceo, "High-Performance Asynchronous Pipeline Circuits," *Proceedings IEEE Second Intern. Symp. on Advanced Research in Asynchronous Circuits and Systems*, 1996.

[YBV+98] K. Yun, P. Beerel, V. Vakilotojar *et al.*, "The Design and Verification of a High-Performance Low-Control-Overhead Asynchronous Differential Equation Solver," *IEEE Trans. on VLSI*, vol. 6, pp. 643–655, Dec. 1998.

[YSK+89] T. Yamasaki, K. Shima, S. Komori *et al.*, "VLSI Implementation of a Variable-Length Pipeline Scheme for Data-Driven Processors," *IEEE J. of Solid-State Circuits*, vol. 24, pp. 933–937, Aug. 1989.

[YSL93] J.-R. Yuan, C. Svensson, and P. Larsson, "New Domino Logic Precharged by Clock and Data," *Electronics Letters*, vol. 29, pp. 2188–2189, Dec. 1993.

[YWG05] S. Yang, B. Winters, and M. Greenstreet, "Energy Efficient Surfing," *Proceedings IEEE Intern. Symp. on Advanced Research in Asynchronous Circuits and Systems*, pp. 2–11, Mar. 2005.